UNIVERSIDAD CARLOS III DE MADRID

Proyecto Fin de Carrera

DESARROLLO DE UNA APLICACIÓN ESCALABLE E INTEROPERABLE PARA LA BÚSQUEDA DE CONTENIDOS EN REDES SOCIALES

Ingeniería Técnica en Informática de Gestión

Departamento de Informática

Autor: Luis Enrique Fabián Lebrón 100060975

Tutor: Pablo Alejandro Acuña

Leganés, Julio de 2010

Título: Desarrollo de una aplicación escalable e interoperable para la búsqueda de contenidos en redes sociales

Autor: Luis Enrique Fabián Lebrón

Director: Pablo Alejandro Acuña

EL TRIBUNAL

Presidente: Susana Montero

Vocal: Juan Diego Álvarez

Secretario: Mario Rafael Ruiz Vargas

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 15 de Julio de 2010 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Me gustaría aprovechar este espacio para agradecer, en primer lugar a mi tutor Pablo Alejandro Acuña, por el trabajo realizado y la ayuda proporcionada a lo largo del desarrollo del proyecto.

También me gustaría mostrar mi agradecimiento a todo el equipo docente de la universidad con quien he tenido ocasión de tratar. De cada uno de ellos he aprendido algo. Mención especial para Daniel Cáceres, antiguo profesor de física en la universidad y con quien mantuve una gran relación.

Por otra parte me gustaría agradecer a mi familia y amigos, en especial a mis padres, mis hermanos, y mi novia el apoyo que siempre me han dado para seguir adelante en mi carrera.

Por último, me gustaría agradecer y dedicar este trabajo de forma especial a mis abuelos.

Resumen

El siguiente trabajo consiste en una aplicación que permite realizar búsquedas de forma simultánea en varias redes sociales, de forma personalizada.

A su vez se caracteriza por ser escalable e interoperable, proporcionando una salida en formato estándar (XML).

También se ha desarrollado una aplicación de ejemplo que utiliza los datos de salida para presentar los resultados en una interfaz gráfica que también permite filtrar los resultados en función de varios criterios y acceder a la información (videos, imágenes o texto) sin salir de la aplicación.

Abstract

The following work is an application that provides to the users the possibility to perform personalized searches through different social networks at the same time.

Furthermore, this application can grow up and its output format is XML, which can be read by different devices.

In addition, an example Website has been developed. It uses the XML document that can be obtained from the search module to show the results through a graphical user interface, which also can filter the results attending to different criteria and access to the information without leaving the application.

Índice de contenidos

Índice de figuras

Índice de tablas

1.-Introducción

Las redes sociales forman cada vez más una parte fundamental dentro de la sociedad en la que vivimos. Nuestra forma de compartir contenido, comunicarnos e interactuar con personas en nuestro entorno real y virtual ha cambiado significativamente en los últimos años.

Con un número de usuarios que aumenta cada día, las redes sociales constituyen cada vez más un nuevo medio de comunicación comparable con otros medios cotidianos como la radio o la televisión. Al momento de escribir esta memoria, la red social Twitter reporta un aproximado de 500 millones de *tweets* (mensajes cortos) al día, un promedio de 600 mensajes cada segundo [6]; Facebook[1] recibe aproximadamente 700 actualizaciones por segundo, y Google Buzz[2], el más reciente de éstos, 55 actualizaciones por segundo [7].

Actualmente podemos encontrar una amplia variedad de redes sociales que permiten a los usuarios compartir contenido en diversos formatos, así como decidir la forma de organizarlos y compartirlos (en modo privado, a un grupo de contactos específico o abierto al público).

Esta cantidad de información actualizada cada segundo puede resultar excesiva al momento de buscar datos acerca de un tema o concepto específico, especialmente si la organización de dicha información es desordenada y no permite categorizar los datos para un uso posterior.

El objetivo principal que persigue este proyecto es desarrollar una herramienta capaz de integrar en un mismo sitio varias redes sociales en las que realizar una búsqueda de contenidos de forma personalizada. A su vez y dado que el número de redes aumenta continuamente, se busca desarrollar una aplicación escalable que permita añadir nuevas redes sociales y tipos de contenido de un modo sencillo. De la misma manera, se utiliza un formato de salida estándar para permitir una interoperabilidad entre diferentes tipos de sistemas, pudiendo utilizarse en aplicaciones para ordenadores, móviles, PDA u otros dispositivos.

1.1.- Objetivos

- Desarrollar una aplicación escalable e interoperable que realice búsquedas personalizadas en diversas redes sociales.

El elemento diferenciador que se busca con este proyecto es la capacidad de dotar al sistema de una estructura que le permita ser *escalable* para poder añadir o quitar redes sociales respecto a necesidades específicas. Al mismo tiempo, se busca proporcionar una salida estándar que permita la *interoperabilidad*, entendiendo ésta

[1] http://www.facebook.com/

[2] http://www.google.com/buzz/

como la capacidad de los sistemas de compartir datos y posibilitar el intercambio de información y conocimiento entre ellos [9].

- Recuperar la información más actual disponible de todas las fuentes disponibles.

Se pretende proveer al usuario final de la información más actual disponible en el conjunto de redes sociales consultadas, al tratarse de escenarios en constante cambio.

- Proporcionar una salida de datos que habilite el desarrollo de nuevos sistemas, utilizando un formato estándar.

Uno de los propósitos principales de este proyecto es crear una herramienta que pueda ser de utilidad para un desarrollador que desee utilizar estos datos de salida en una aplicación, ya sea para un teléfono móvil, una televisión o cualquier otro dispositivo con acceso a Internet. Para esto, los datos de salida deben tener un formato estándar para maximizar la compatibilidad.

- Permitir al usuario final personalizar los criterios de su búsqueda.

Tanto para un desarrollador como para un usuario final, se plantea permitir la personalización de las redes sociales de las cuales obtener información, así como el número específico de resultados a devolver.

- Diseñar un sistema abierto y extensible a diversas redes sociales.

Se busca dotar a la aplicación de una estructura de datos que permita ampliar su funcionalidad para dar soporte a diversas redes sociales.

- Desarrollar una aplicación que haga uso de la salida del módulo de búsqueda.

Para dar una idea de las diferentes posibilidades que proporciona la salida estándar obtenida, se creará una aplicación Web que permitirá al usuario utilizar las principales funcionalidades aquí descritas, así como permitir filtrar los datos y acceder a la información ofrecida de acuerdo a sus preferencias.

1.2.- Descripción del documento

En esta sección se detallan los contenidos de cada uno de los capítulos de este documento.

-Descripción del problema.

Se puede encontrar la información relativa al problema a solucionar, así como alternativas existentes actualmente y las características que diferencian este proyecto de las demás.

-Metodología.

En este capítulo se detallará toda la información relativa a las diferentes metodologías y paradigmas de desarrollo utilizados en el proyecto, así como la planificación temporal del mismo mediante un diagrama de Gantt.

-Estado del arte.

Se describen las distintas tecnologías y protocolos que se han utilizado durante la implementación de este proyecto.

-Análisis.

Este capítulo está dedicado a analizar el problema a resolver, establecer los requisitos del sistema y su finalidad.

-Diseño.

Capítulo en el cual se describe el diseño realizado previo a la implementación de la aplicación, incluyendo diversos diagramas UML que describen el comportamiento que se busca en el sistema.

-Implementación

En este capítulo se detallan los pasos que siguen tanto el módulo de búsqueda como la aplicación de ejemplo para cumplir los objetivos.

-Caso de uso

Esta sección refleja un ejemplo de uso de la aplicación, adjuntando figuras e ilustraciones que permiten comprender cómo utilizar el sistema. Además se realizará una comparativa con los datos originales obtenidos directamente en cada una de las páginas de las redes sociales incluidas.

-Evaluación.

En este capítulo se detalla específicamente el proceso para ampliar el módulo de búsqueda, proporcionando funcionalidad para una nueva red social.

-Conclusiones.

Contiene las diferentes impresiones y opiniones una vez concluido el desarrollo del proyecto.

-Anexos.

Contienen información útil para entender algunos ejemplos, incluyendo documentos en formato XML utilizados en los capítulos del documento.

-Bibliografía y referencias.

Conjunto de fuentes y publicaciones que han servido como referencia a la hora de obtener el conocimiento relativo tanto al problema a resolver como a las tecnologías a utilizar.

2.-Descripción del Problema

En esta sección se describe el problema afrontado en este proyecto, así como la solución planteada para resolverlo. Adicionalmente, se describe a grandes rasgos los componentes del sistema a modo de introducción para las secciones siguientes.

El problema que decidimos afrontar es encontrar una manera sencilla y eficaz de buscar información publicada por los usuarios de redes sociales, de una forma escalable y en la que destaque la portabilidad para permitir cubrir las necesidades de los usuarios actualmente.

El mayor problema al que nos enfrentamos es la gran cantidad de información distribuida entre diferentes redes sociales con diferentes formatos: textos, imágenes, vídeos, etc.; y la ausencia de un único sitio en el que realizar búsquedas de manera personalizada a estos sitios. Con este proyecto buscamos proporcionar una herramienta que actúe de intermediario entre el contenido almacenado en las redes sociales y los usuarios en busca de dicho contenido utilizando una gran variedad de dispositivos potenciales en los que interactuar con los resultados.

A grandes rasgos, el proyecto consta de dos componentes principales: un módulo de búsqueda y una aplicación ejemplo. El componente más importante, que ofrece la solución al problema descrito, es un módulo de búsqueda independiente (“*stand-alone”*) que genera un documento estándar con los resultados obtenidos; el segundo componente es una aplicación ejemplo que demuestra la funcionalidad del módulo de búsqueda.

El módulo de búsqueda entrará en contacto con las API’s[3] de cada una de las redes sociales consideradas para este proyecto. Nuestra aplicación principal realizará solicitudes a las API’s correspondientes y obtendrá un resultado, que variará en función de las redes sociales con la que esté trabajando en cada momento.

Una vez se han recibido los resultados, deben tratarse para integrarlos en un mismo documento estándar en formato XML[4], el cual contendrá un compendio de todos los resultados encontrados en las redes sociales buscadas. El estándar XML permite proveer una alta portabilidad de la información generada [8].

Llegados a este punto, la aplicación Web ejemplo leerá el documento XML que ha generado la aplicación, presentando los resultados correspondientes. Como se ha comentado en anteriores apartados, el hecho de que el formato de salida sea XML, nos permite tener un resultado altamente portable, ya que también puede ser leído por una aplicación en un móvil, en una PDA, entre otros dispositivos.

---

[3] Application Program Interface: se trata de un conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software como una capa de abstracción. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio.

[4] XML: eXtensible Markup Language

En la siguiente figura podemos ver una representación gráfica de la aplicación que se plantea desarrollar.

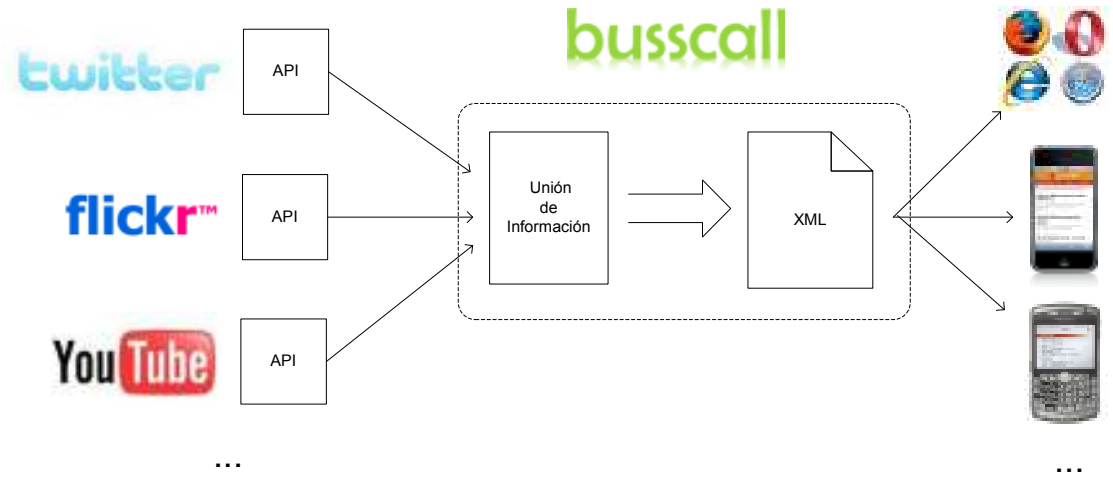

Ilustración 1: Descripción del sistema

Como podemos ver en la anterior ilustración, *busscall*, como hemos denominado a este proyecto, se conectará con el API de cada una de las redes solicitadas en cada búsqueda, y uniendo todos los datos recolectados generará un documento XML. Dicho documento, se puede tratar para múltiples propósitos, desde crear una aplicación Web que muestre los resultados obtenidos, como es nuestro caso, o una aplicación para iPhone o Blackberry con la misma o diferente finalidad.

Con el desarrollo de este proyecto, se tiene como objetivo crear una plataforma capaz de utilizar distintas redes sociales y que pueda ampliarse de una forma sencilla para el desarrollador. Se seleccionaron las redes Flickr, YouTube y Twitter, dada la importancia percibida de las mismas y la variedad en el formato de contenidos que proporcionan, al trabajar cada una de ellas con imágenes, vídeos y texto, respectivamente.

A continuación, vamos a explicar las características de las principales redes sociales con las que este proyecto tiene relación:

Twitter

La red Twitter[5] nació como un proyecto de investigación y desarrollo a comienzos del año 2006. Se trata de un servicio gratuito de *microblogging* que permite el envío de pequeños mensajes con un máximo de 140 caracteres conocidos como 'tweets', los cuales pueden ser publicados y compartidos desde la página principal o desde una amplia variedad de aplicaciones externas, incluyendo aplicaciones Web

[5] http://www.twitter.com/

como Facebook o aplicaciones para ordenadores y móviles como Tweetie[6] o HootSuite[7].

Cuando un usuario publica una nueva entrada, aparece en su perfil de usuario y además son enviadas de forma inmediata a los usuarios que hayan elegido recibirlas, es decir, que se hayan suscrito como "seguidores" de su perfil.

Twitter permite configurar la privacidad de las publicaciones, de forma que permite el acceso a los mensajes de manera pública (es la opción por defecto), o poder restringirlas sólo a los seguidores. Además, Twitter permite a los usuarios consultar actualizaciones por una gran cantidad de canales diferentes, como SMS, e-mail, etc.

Se trata de una red social que se caracteriza por su simplicidad, sólo almacena pequeños mensajes de texto, los cuales pueden incluir enlaces a imágenes o a otro contenido. Es utilizada cada vez más por empresas y personajes famosos como un medio de comunicación con sus seguidores.

Flickr

Otra de las redes sociales que integraremos en este proyecto es Flickr[8]. Nació en 2004 y en poco más de un año pasó a formar parte del grupo Yahoo!™. Esta red social se centra en la posibilidad de compartir fotografías, aunque también nos permite subir vídeos cortos. Cada imagen puede estar asociada a un número indeterminado de etiquetas o 'tags' que la describen, así como permite etiquetar usuarios de Flickr. Flickr permite interacción a través de grupos, contactos directos o 'amigos', creación de galerías, entre otras cosas.

En Flickr la privacidad es importante, y permite configurar el acceso que los demás usuarios tienen a los videos o imágenes que se publican, de forma que estas pueden ser públicas o privadas, dentro de la cual podemos distinguir entre 'familiares' y 'amigos' en general.

Flickr es una de las redes sociales más utilizadas en la actualidad y resulta de gran interés su inclusión en este proyecto ya que permite obtener contenidos en distintos formatos (almacena tanto vídeos como imágenes), utiliza un sistema de etiquetas para describir el contenido además de incluir otras características interesantes como geo-ubicación. Finalmente, proporciona un API muy potente que permite obtener el contenido a través de diferentes medios y en diversos formatos.

YouTube

Finalmente, YouTube[9] es sin duda una de las redes sociales más conocidas y utilizadas en todo el planeta, nació en 2005 y en menos de 10 meses fue comprada por

---

[6] http://www.atebits.com/tweetie-iphone/

[7] http://hootsuite.com/

[8] http://www.flickr.com/

[9] http://www.youtube.com/

el grupo Google™ por un montante aproximado de 600 millones de dólares (después de lanzar sus propias aplicaciones sin éxito).

YouTube permite subir videos a la aplicación, así como clasificarlos y compartirlos; podemos fácilmente colocar un vídeo de YouTube en nuestra propia página Web, foros, etc. Los usuarios pueden comentar los vídeos publicados, crear sus propios canales, suscribirse a los vídeos publicados por un determinado autor, valorar un vídeo, etcétera.

En los últimos meses, YouTube ha ampliado sus características permitiendo publicar videos en alta definición, con el aumento de calidad que esto implica. Además, está integrada con multitud de plataformas que permiten publicar videos de forma sencilla y rápida (v.g. el Iphone).

YouTube es una de las páginas Web más visitadas en el mundo a diario [10]. Resulta de interés que la aplicación Busscall sea compatible con esta red social ya que es la que aglutina el mayor porcentaje de videos disponibles en Internet. Además proporciona información adicional como etiquetas y categorías, y dispone de canales y medios para que los distintos usuarios interaccionen entre sí.

Se ha decidido utilizar las tres redes sociales explicadas anteriormente debido a la relevancia de las mismas, y la posibilidad de manejar diferentes tipos de contenidos, imagen, vídeo y texto. Cada una de las redes sociales seleccionadas atrae suficiente atención en el mundo de Internet para encontrar y publicar vídeos (Youtube), fotografías (Flickr) y texto (Twitter).

Finalmente, debemos mencionar algunas aplicaciones Web cuyas características tienen relación con el proyecto a realizar ya que su funcionalidad es similar a la que se pretende dotar a este proyecto:

Tumblr[10]

Esta red social nació en 2007 y su crecimiento es cada vez mayor. Entre los meses de Enero y Junio, el tráfico de esta red social ha estado cerca de triplicarse [13], lo que puede proporcionar una idea de su imparable evolución.

Tumblr permite al usuario compartir todo tipo de contenidos, ya sea texto, imágenes, videos, etc. A su vez, integra varias redes sociales como Facebook o Twitter, y permite publicar contenido en ellas.

Contrastando Tumblr con el proyecto que se describe en este documento, tienen en común varias características, ya que Tumblr permite integrar varias redes sociales en una única aplicación y proporciona funciones adicionales, como por ejemplo la publicación de contenido.

Nuestro proyecto tiene un propósito más concreto, y es permitir al usuario personalizar una búsqueda entre diferentes redes sociales, y proporcionar una salida

[10] http://www.tumblr.com/

que se pueda utilizar en diferentes dispositivos, es decir, su propósito es ser útil de cara a diferentes desarrolladores. Tumblr proporciona directamente un servicio a los usuarios a través de una aplicación Web completa.

FriendFeed[11]

Se trata de una red social muy parecida a Tumblr. Permite a los usuarios integrar diferentes redes sociales, de forma que pueden seguir los contenidos que publican sus amigos de redes sociales de todo tipo, abarcando desde contenido multimedia como imágenes o videos, hasta redes sociales de ámbito profesional como LinkedIn [12]o Xing[13].

FriendFeed proporciona a la vez de la posibilidad de publicar contenido en diferentes redes sociales como la de realizar búsquedas en todas ellas, pero de igual forma que Tumblr, es un servicio que no permite personalizar dicha búsqueda, especificando qué redes concretas interesan al usuario, cuantos datos se desea obtener, etc. Por otra parte, sus servicios se utilizan dentro de la misma aplicación, y no está orientada a proporcionar una salida estándar que pueda ser reutilizada con diversas finalidades, como es el caso del proyecto que se plantea.

En definitiva, podemos afirmar que estas alternativas proporcionan un servicio concreto a los usuarios a través de su aplicación Web, mientras que nuestro proyecto se centra en la escalabilidad, la posibilidad de personalizar una búsqueda y proporcionar una salida única en un formato estándar que permita reutilizar los datos por parte de otros desarrolladores que pretendan crear proyectos de diferente índole o dirigidos a otros dispositivos.

[11] http://www.friendfeed.com

[12] http://www.linkedin.com

[13] http://www.xing.com

3.- Metodología

Para desarrollar este proyecto hemos decidido emplear una metodología basada en un **ciclo de vida evolutivo** [1]**.** El desarrollo evolutivo se basa en la idea de realizar una implementación inicial e irla refinando hasta que se obtiene un sistema que se considere adecuado. Las actividades de desarrollo y validación se entrelazan en lugar de separarse, y a su vez, existe una rápida retroalimentación entre ambas. Este modelo acepta que los requerimientos del usuario puedan cambiar en cualquier momento.

La práctica nos demuestra que obtener todos los requerimientos al comienzo del proyecto es extremadamente difícil, no sólo por la dificultad del usuario de transmitir su idea, sino porque estos requerimientos evolucionan durante el desarrollo y de esta manera, surgen nuevos requerimientos a cumplir. El modelo de ciclo de vida evolutivo afronta este problema mediante una iteración de ciclos en los cuales se incluyen posibles nuevos requisitos, posteriormente se mejora o cambia el desarrollo y por último se lleva a cabo una nueva evaluación.

En la siguiente ilustración [1] podemos ver de forma más clara como se ha ido desarrollando este proyecto.

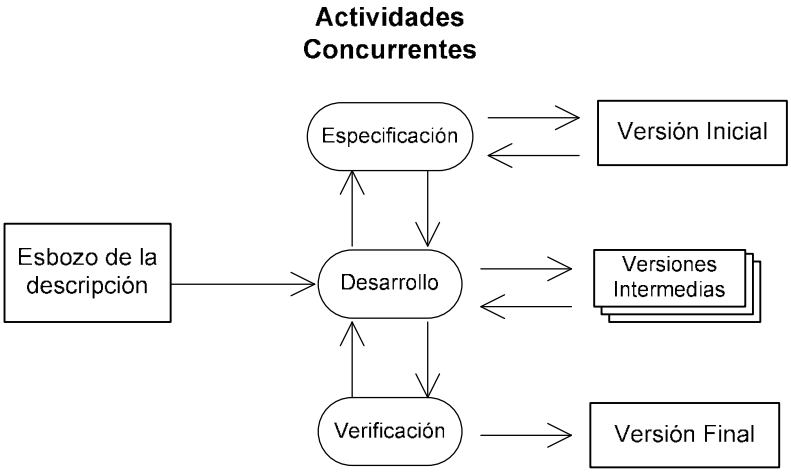

Ilustración 2: Desarrollo evolutivo

Se ha aplicado la metodología descrita anteriormente para desarrollar este proyecto. Para ello, en un primer esbozo de la solución se plantearon distintas alternativas para poder llevar a cabo las búsquedas, planteando distintos criterios de búsqueda, como aplicar una búsqueda por tipo de contenidos – llegando a la conclusión de que no existe una forma sencilla y eficaz de llevar a cabo este tipo de búsqueda dada la variabilidad de las redes sociales – por lo que finalmente se decidió realizar la búsqueda por cada una de las redes sociales y no por el tipo de contenido a obtener.

Durante la especificación, se decidió implementar este proyecto con tres redes sociales que contengan contenidos en distintos formatos, y que a su vez sean redes sociales de relevancia en Internet. A su vez, se definió la estructura general del documento XML que la aplicación proporciona como salida.

El primer desarrollo del proyecto, se centró en obtener un primer prototipo que garantizase la escalabilidad y estabilidad del sistema, de forma que añadir nuevas redes sociales sea sencillo y no comprometa en ningún momento el funcionamiento del sistema. Posteriormente, se verificó que esta primera versión funcionaba correctamente y Busscall comenzó a evolucionar.

Durante la segunda etapa de desarrollo, se implementó la primera clase que trabajaba directamente con una red social, en este caso la primera fue Twitter ya que presentaba un contenido muy simple. Una vez verificado, se fueron incluyendo progresivamente las otras dos redes sociales: Flickr y YouTube.

A medida que el proyecto iba creciendo, la primera especificación del documento XML de salida quedó obsoleta por lo que se fue actualizando la misma, con nuevos campos que resultaban de interés almacenar para redes sociales que proporcionan mayor información que Twitter.

Una vez implementadas todas las redes sociales, se procedió a verificar el funcionamiento de la aplicación, obteniendo así la versión final del mismo.

Por otra parte, también es necesario destacar el patrón de diseño que se ha seguido durante el proyecto. Se trata del patrón conocido como MVC (*Model-View-Controller* o Modelo-Vista-Controlador [11]).

MVC define una arquitectura de software en la cual se separan los datos de una aplicación, su interfaz de usuario y la lógica de control en tres componentes diferentes. Modelo-Vista-Controlador, es muy utilizado en el desarrollo de aplicaciones Web. A continuación se describen las capas que componen MVC:

-**Modelo:** Esta capa es la representación específica de la información con la cual el sistema opera.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

En resumen, el modelo se limita a lo relativo de la vista y su controlador facilitando las presentaciones visuales complejas. El sistema también puede operar con más datos no relativos a la presentación, haciendo uso integrado de otras lógicas de negocio y de datos afines con el sistema modelado.

-**Vista:** Presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Las vistas son responsables de:

- Recibir los datos del modelo o eventos del controlador y mostrarlos al usuario.

-**Controlador:** Capa encargada de responder a eventos, usualmente acciones del usuario, invoca peticiones al modelo y, a la vista.

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos del tipo: "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

En la siguiente ilustración [12], puede ver se una descripción gráfica de cómo funciona MVC.

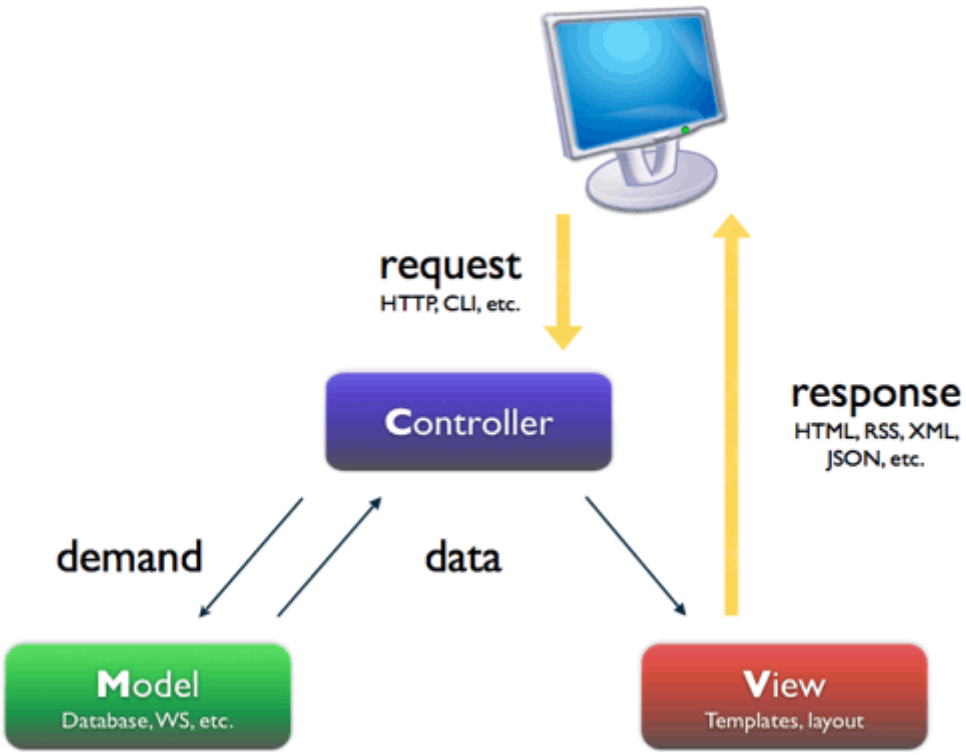

Ilustración 3: Modelo Model-View-Controler

Una vez descritas las capas que conforman el patrón, resulta necesario explicar qué parte de este proyecto corresponde con cada una de ellas.

El Modelo en este proyecto está formado por las diferentes clases que permiten conectar con las API de las distintas redes sociales soportadas. Las diferentes clases del modelo se ponen en contacto con el controlador, que será el encargado de gestionar las vistas.

La capa de Controlador es asumida por el módulo de búsqueda, el cual se encarga dinámicamente de invocar a las diferentes clases que componen el modelo en función de la búsqueda a realizar. Es decir, como aparece reflejado en la figura anteriormente señalada, realiza una solicitud (*demand*), para posteriormente recibir un conjunto de datos (*data*). Una vez se ha recibido dicha información, el controlador entra en contacto con la capa View (Vista) para ofrecer al usuario los resultados.

Por último, la capa Vista está formada en este proyecto por la aplicación de ejemplo desarrollada, la cual recibe los datos del controlador incluyendo tanto los resultados de una búsqueda como los parámetros deseados por el usuario (por ejemplo, el criterio de filtrado de datos). Sobre dichos datos, esta capa proporciona al usuario una interfaz mediante la cual puede interactuar con la aplicación.

Planificación

Como se ha indicado anteriormente, seguiremos un ciclo de vida evolutivo, por lo que algunas fases se realizan en paralelo en varias ocasiones. En primer lugar, se debe investigar acerca de las tecnologías a utilizar y sobre el estado del arte, para ello se destinará el mes de Diciembre de 2009 de forma exclusiva.

Una vez se conoce el entorno del problema, se define y especifica el mismo detallando al máximo las características del módulo a implementar, como los requisitos necesarios para su correcto funcionamiento (servidores, software, entornos de desarrollo, etc.). Para completar estas tareas se reservan 30 días de trabajo.

Una vez llegados a este punto, comienza el desarrollo del sistema. De forma iterativa y flexible, entre los meses de Febrero y Mayo, se lleva a cabo el análisis de requisitos, el diseño conceptual y específico, así como la implementación y la evaluación de los diferentes prototipos.

Finalizado el desarrollo, se completa la documentación del proyecto y se lleva a cabo la validación del mismo mediante pruebas específicas realizadas sobre la aplicación. Estas dos tareas también se realizan en paralelo y para ello se utilizará todo el mes de Junio.

El día 30 de Junio, se establece como la fecha límite para finalizar todas las tareas, de cara a la entrega final del proyecto alrededor del 14 de Julio. No obstante, se establecen dos semanas de margen para evitar imprevistos y disponer de un plazo de tiempo razonable para afrontar retrasos inesperados en las tareas a realizar.

En la siguiente página puede verse un diagrama de Gantt con la planificación temporal completa para este proyecto.

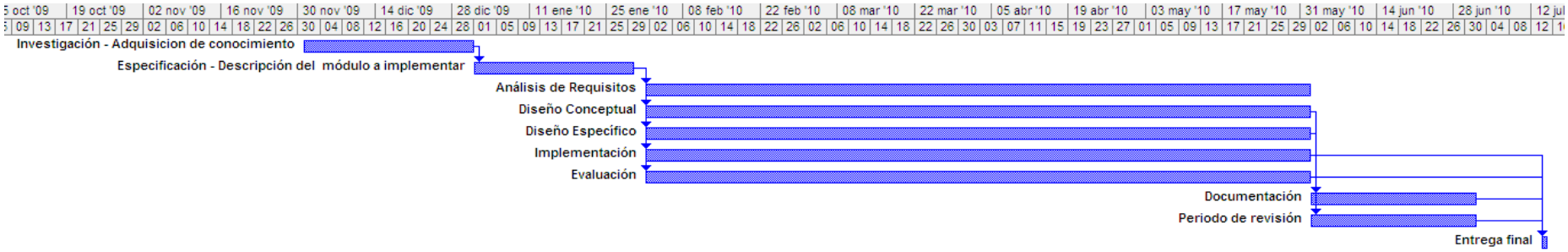

Ilustración 4: Diagrama de Gantt

	Nombre de tarea	Comienzo	Fin	Duración
1	Investigación - Adquisicion de conocimiento	mar 01/12/09	jue 31/12/09	23 días
2	Especificación - Descripción del módulo a implementar	vie 01/01/10	vie 29/01/10	21 días
3	Análisis de Requisitos	lun 01/02/10	mar 01/06/10	87 días
4	Diseño Conceptual	lun 01/02/10	mar 01/06/10	87 días
5	Diseño Específico	lun 01/02/10	mar 01/06/10	87 días
6	Implementación	lun 01/02/10	mar 01/06/10	87 días
7	Evaluación	lun 01/02/10	mar 01/06/10	87 días
8	Documentación	mié 02/06/10	jue 01/07/10	22 días
9	Periodo de revisión	mié 02/06/10	jue 01/07/10	22 días
10	Entrega final	mié 14/07/10	mié 14/07/10	1 día

Ilustración 5: Lista de tareas

PRESUPUESTO DE PROYECTO

1.- Autor: Luis Enrique Fabián Lebrón

2.- Departamento: Informática

3.- Descripción del Proyecto:

- Titulo: Busscall
- Duración (meses): 6

Tasa de costes Indirectos: **10%**

4.- Presupuesto total (valores en Euros):

19.085,00 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes)	Coste hombre mes	Coste (Euro)
Luis Enrique Fabián Lebrón		Ingeniero	131,25	1.750,00	0,00 10.500,00
		Hombres mes 1		**Total**	10.500,00

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Equipo de desarrollo	780,00	100	6	60	6.850,00
				Total	6.850,00

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	10.500
Amortización	6.850
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	1.735
Total	19.085

4.- Estado del arte

4.1.- Introducción

A través de este capítulo, llevaremos a cabo un estudio de las tecnologías disponibles que pueden ser utilizadas para llevar a cabo el desarrollo de este proyecto. Para ello, se ha llevado a cabo un proceso de investigación en el cual se han recopilado fuentes de información de distinta índole, desde libros a diferentes sitios Web relacionados con las tecnologías utilizadas.

En esta sección se puede encontrar una descripción del lenguaje XML y muchas tecnologías asociadas para procesarlos como XSL o XPath, así como los analizadores sintácticos para este formato que están disponibles para PHP, que será el lenguaje de programación en el cual está implementado este proyecto. A su vez, se incluyen otros formatos relacionados con XML como RSS o Atom y algunas alternativas a XML que también serán utilizadas durante el proyecto como JSON.

Además de toda la información relacionada con XML, se encontrará una descripción acerca de los *Web Services*, y se introducirán conceptos relacionados como SOAP o REST, utilizados durante el desarrollo de Busscall.

4.2.- XML

XML (siglas en ingles de eXtensible *Markup Language*[8] *o* lenguaje de marcas extensible, en español), es un meta-lenguaje desarrollado por el World Wide Web Consortium (W3C[14]). Por lo tanto, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML, etc. En definitiva, XML no es, como su nombre podría sugerir, un lenguaje de marcado, si no un meta-lenguaje que permite definir lenguajes de marcado adecuados a determinados usos.

XML es un elemento fundamental dentro de nuestro proyecto, ya que será el lenguaje estándar en el cual obtendremos los resultados de las búsquedas que llevemos a cabo en las diferentes redes sociales. En este apartado, veremos tanto la historia de este lenguaje como la estructura de un documento XML bien formado, y de otros elementos relacionados como DTD's, Namespaces…

4.2.1.-Historia de XML

XML proviene de un lenguaje inventado por IBM en los años setenta, llamado GML (*Generalized Markup Language*), que surgió por la necesidad que tenía la empresa de almacenar grandes cantidades de información. Este lenguaje gustó a la ISO[15], por lo que en 1986 trabajaron para normalizarlo creando el lenguaje SGML (*Standard Generalized Markup Language*), capaz de adaptarse a un gran abanico de problemas. A partir de él se han creado otros lenguajes para almacenar información.

En el año 1989 Tim Berners-Lee creó la Web, y junto a ella el lenguaje HTML. Este lenguaje se definió en el marco de SGML y fue de lejos la aplicación más conocida de este estándar. Sin embargo, los navegadores Web siempre han puesto pocas exigencias al código HTML que interpretan, y así las páginas Web son caóticas y en ocasiones no cumplen correctamente con su sintaxis. Estas páginas Web dependen fuertemente de una forma específica de lidiar con los errores y las ambigüedades, lo que hace a las páginas más frágiles y a los navegadores más complejos.

Otra limitación del HTML es que cada documento pertenece a un vocabulario fijo. No se pueden combinar elementos de diferentes vocabularios. Asimismo, es imposible para un intérprete (por ejemplo un navegador) analizar el documento sin tener conocimiento de su gramática. Por ejemplo, el navegador sabe que antes de una etiqueta debe haberse cerrado cualquier <p> previamente abierto. Los navegadores resolvieron esto incluyendo lógica *ad-hoc* para HTML, en vez de incluir un analizador genérico. Ambas opciones, de todos modos, son muy complejas para los navegadores.

[14] http://www.w3.org - http://www.w3c.es

[15] http://www.iso.org

Se buscó entonces definir un subconjunto del SGML que permitiese:

- Mezclar elementos de diferentes lenguajes. Es decir que los lenguajes sean extensibles.
- La creación de analizadores sintácticos simples, sin ninguna lógica especial para cada lenguaje.
- Empezar de cero y hacer hincapié en que no se acepte nunca un documento con errores de sintaxis.

Para hacer esto XML deja de lado muchas características de SGML que estaban pensadas para facilitar la escritura manual de documentos. XML en cambio está orientado a hacer las cosas más sencillas para los programas automáticos que necesiten interpretar el documento.

4.2.2.-Ventajas de XML

XML se creó para cumplir varios objetivos, como ser idéntico a la hora de enviar, recibir y procesar la información del mismo modo que HTML, para aprovechar toda la tecnología implantada para este último. Además es formal y conciso desde el punto de vista de los datos y la manera de guardarlos. También es extensible, así como fácil de leer editar y programar. XML se puede usar para una gran variedad de trabajos y aporta muchas ventajas en amplios escenarios. Algunas ventajas de este lenguaje en algunos campos prácticos son:

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador sintáctico es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan errores (*bugs*) y se acelera el desarrollo de aplicaciones.

Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla.

XML mejora la compatibilidad entre aplicaciones como se detalla a continuación:

- Comunicación de datos. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.
- Migración de datos: Si tenemos que mover los datos de una base de datos a otra sería muy sencillo si las dos trabajasen en formato XML.
- Aplicaciones Web: Hasta ahora cada navegador interpreta la información a su manera y los programadores Web deben variar su forma de trabajar en función del navegador del usuario. Con XML tenemos una sola aplicación que maneja los

datos y para cada navegador podremos tener, por ejemplo una hoja de estilos para aplicarle un estilo adecuado. Si en un futuro nuestra aplicación debe ejecutarse en dispositivos móviles sólo tendremos que crear una nueva hoja de estilo.

4.2.3.- Estructura de un documento XML

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Consecuentemente, tenemos un arbol estructurado con nodos de información. Por ejemplo, los resultados de una búsqueda, pueden estar compuestos por un título, un enlace, una fecha, una descripción, etc. Estas partes se llaman elementos, y se las señala mediante etiquetas.

Una etiqueta consiste en una *marca* hecha en el documento, que señala una parte de éste como un elemento. Una información con un sentido claro y definido. Las etiquetas tienen la forma *<nombre>*, donde 'nombre' es el nombre del elemento que se está señalando.

Veamos un ejemplo sencillo de un documento XML así como su estructura:

```
<?xml version="1.0"?>
<!DOCTYPE MENSAJE SYSTEM "busqueda.dtd">
<resultado>
  <remite>
      <redsocial>YouTube</redsocial>
      <enlace>http://www.YouTube.com/watch?v=LU8DDYz68kM</enlace>
  </remite>
 <contenido>
    <titulo>Battle at Kruger</titulo>
    <fecha>03-05-2007</fecha>
    <descripcion>A battle between a pride of lions, a herd of buffalo, and 2 crocodiles at a watering hole
in South Africa's Kruger National Park while on safari. </descripcion>
 </contenido>
</resultado>
```

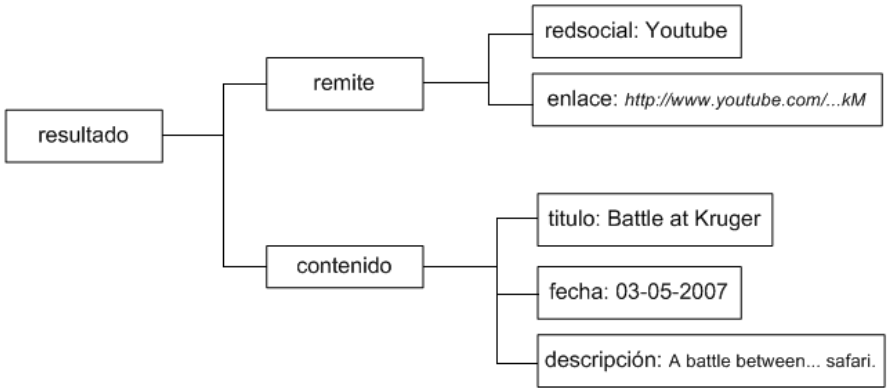

Ilustración 6: Ejemplo de XML

Como podemos ver en el ejemplo XML, que se corresponde con la figura [14], tenemos un resultado de YouTube, compuesto por dos elementos, remite y contenido. El primero de ellos almacena la red social y un enlace al vídeo, mientras que el segundo contiene el título, la fecha de subida y una descripción del mismo.

4.2.3.1.- Document Type Definitions (DTDs)

Una definición de tipo de documento o DTD [17] (siglas en inglés de *Document Type Definition*) es una descripción de estructura y sintaxis de un documento XML o SGML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. De esta forma, dichos documentos pueden ser validados a partir de la estructura de los elementos y la descripción de los datos que incluye cada documento, y pueden además compartir la misma descripción y forma de validación dentro de un grupo de trabajo que usa el mismo tipo de información.

Los DTD se utilizan normalmente para determinar la estructura de un documento mediante etiquetas XML o SGML. Un DTD describe:

- Elementos: indican qué etiquetas están permitidas y su contenido.
- Estructura: indica el orden de las etiquetas en el documento.
- Anidamiento: indica qué etiquetas pueden incluirse dentro de otras.

Veamos un ejemplo de DTD relacionado con el XML de ejemplo propuesto anteriormente:

```
<!DOCTYPE busqueda [
        <!ELEMENT resultado (remite,contenido)>
        <!ELEMENT remite      (redsocial,enlace)>
        <!ELEMENT redsocial    (#PCDATA)>
        <!ELEMENT enlace (#PCDATA)>
        <!ELEMENT contenido  (titulo,fecha, descripcion)>
        <!ELEMENT titulo    (#PCDATA)>
        <!ELEMENT fecha (#PCDATA)>
        <!ELEMENT descripcion (#PCDATA)>
        ]>
```

4.2.3.2.- XML Schema

Esta tecnología [18], se presenta como una alternativa frente a los DTD's, para definir la estructura de un documento XML. La ventaja de los *Schemas* con respecto a los documentos DTD's son:

- Usan sintaxis de XML, al contrario que los DTDs.
- Permiten especificar los tipos de datos.
- Son extensibles.
- Garantiza documentos XML válidos y bien formados.

De forma análoga al apartado anterior, podemos ver en la siguiente tabla, el Schema del mismo ejemplo [19].

```
<?xml version="1.0"?>
        <xs:schemaxmlns:xs="http://…"
                targetNamespace="http://…"
                xmlns="http://…"
                elementFormDefault="qualified">
        <xs:element name="resultado">
          <xs:complexType>
                  <xs:sequence name=”remite”>
                          <xs:element name="redsocial" type="xs:string"/>
                          <xs:element name="enlace" type="xs:string"/>
                     </xs:sequence>
         <xs:sequence name=”contenido”>
                          <xs:element name="titulo" type="xs:string"/>
                          <xs:element name="fecha" type="xs:date"/>
                          <xs:element name="enlace" type="xs:string"/>
                     </xs:sequence>
                   </xs:complexType>
                   </xs:element>
                  </xs:schema>
```

4.2.3.3.- Namespaces en XML

En general, en informática, un Namespace o espacio de nombres es un medio para organizar clases dentro de un entorno, agrupándolas de un modo más lógico y jerárquico. Por ejemplo, si estamos desarrollando un conjunto de clases para las operaciones de gestión de una biblioteca, podemos ir escribiendo todas las clases y situarlas dentro de una misma aplicación o DLL (biblioteca de enlace dinámico). Supongamos que tenemos dos clases para operaciones de gestión de préstamos, denominadas *Titulo* y *Fecha*, y otras dos clases para operaciones de compra, denominadas *Factura* y *Precio*.

Si necesitáramos añadir una clase más para las compras que registre el título del libro, el nombre más idóneo sería *Titulo*, pero este nombre ya está siendo utilizado, así que para evitar problemas de duplicidad podríamos elegir otro que puede no ajustarse a la funcionalidad de la clase.

Este problema se puede solucionar mediante el uso de Namespaces, que permiten organizar mejor cada clase, ya que podemos asignarle un nombre jerárquico para la funcionalidad que desempeña. De esta forma, podríamos crear un Namespace con el nombre *Gestión*, que contuviera otros dos Namespaces llamados *Prestamo* y *Compra*, para finalmente incluir en cada uno de ellos las clases correspondientes.

El concepto de espacio de nombres en lenguaje XML es similar, pero difiere en dos puntos fundamentales: las versiones de XML no tienen estructura interna y no se trata, matemáticamente hablando, de un conjunto.

Los espacios de nombres permiten eliminar las ambigüedades y solucionar los problemas de homonimia que se producen en los documentos, ya que en un mismo documento podemos encontrar palabras con el mismo nombre (por ejemplo, "capital"), pero con diferentes significados (término geográfico y término económico).

<país nombre="España">
<capital>Madrid</capital>
</país>

<inversión>
<capital>2000€</capital>
</inversión>

4.3.- Procesamiento de XML

4.3.1.- Extensible Stylesheet Language

XSL (siglas en inglés de *Extensible Stylesheet Language* [15], o lenguaje de hojas de estilo ampliable en español) es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML debe ser transformada o formateada para su presentación en un medio.

XSL es para XML lo que las hojas de estilo CSS son para los documentos HTML. Está diseñado para presentar datos XML en un formato legible. XSL consta realmente de dos partes:

- XSLT: un lenguaje para transformar documentos XML.
- XPath: un lenguaje para navegar en documentos XML.

XSLT significa Transformaciones XSL y es la parte más importante de XSL. Puede transformar un documento XML en una página HTML, otro XML o un texto sencillo. El uso de XSL es imprescindible, ya que las etiquetas XML han sido definidas por el usuario y, por tanto, los navegadores no saben cómo interpretarlas o representarlas. Su significado se ha diseñado para ser entendido por las personas, no por las máquinas.

XSLT también puede realizar las siguientes operaciones en un árbol XML:

- Añadir y eliminar elementos
- Añadir y eliminar atributos
- Reorganizar y ordenar elementos
- Ocultar o mostrar determinados elementos
- Encontrar o seleccionar elementos específicos

¿Cómo encaja XSL en el contexto global? [16] Para producir las páginas HTML finales que verán los visitantes del sitio Web, es necesario aplicar una hoja de estilos XSL a la fuente de datos XML. La transformación puede realizarse en el servidor Web o en el navegador del cliente. El resultado final puede ser una página HTML completa o solamente una parte de ella que se utiliza en distintas páginas.

4.3.2.- XPath

Todo el procesamiento realizado con un fichero XML está basado en la posibilidad de direccionar o acceder a cada una de las partes que lo componen, de modo que podamos tratar cada uno de los elementos de forma diferenciada.

XPath[16] (*XML Path Language* [20]) es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos. Permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML. Fue creado para su uso en XSLT, que lo usa para seleccionar y examinar la estructura del documento de entrada de la transformación.

XPath constituye un lenguaje sofisticado y complejo, pero distinto de los lenguajes procedurales comunes (C, C++, Basic, Java). Es a su vez la base sobre la que se han especificado nuevas herramientas para el tratamiento de documentos XML, como XPointer[17], XLink [18]o XQuery[19], lenguaje que maneja los documentos XML como si de una base de datos se tratase.

Así, XPath sirve para poder navegar dentro de un documento XML de forma sencilla, obteniendo así sólo los datos que son requeridos en cada momento.

4.3.3.-XQuery

XQuery [22] es un lenguaje de consulta diseñado para consultar colecciones de datos XML. Es semánticamente similar a SQL, pero incluye algunas capacidades de programación. Proporciona los medios para extraer y manipular información de documentos XML, o de cualquier fuente de datos que pueda ser representada mediante XML, como por ejemplo Bases de Datos Relacionales o documentos ofimáticas.

XQuery utiliza expresiones XPath para acceder a determinadas partes del documento XML. Añade además unas expresiones similares a las usadas en SQL. Incluye la posibilidad de construir nuevos documentos XML a partir de los resultados de la consulta. Se puede usar una sintaxis similar a XML si la estructura (elementos y atributos) es conocida con antelación, o usar expresiones de construcción dinámica de nodos en caso contrario.

XQuery 1.0, por otra parte no incluye capacidad de actualizar los documentos XML. Tampoco puede realizar búsquedas textuales. Estas dos capacidades están siendo objeto de desarrollo para su posible incorporación en la siguiente versión del lenguaje.

---

[16] http://www.w3.org/TR/xpath/

[17] http://www.w3.org/TR/WD-xptr

[18] http://www.w3.org/TR/xlink/

[19] http://www.w3.org/TR/xquery/

4.4.- Aplicaciones de XML: Sindicación de Información

4.4.1.- RSS

RSS (siglas en inglés de *Rich Site Summary* o *Really Simple Syndication* [24]). Es un estándar creado para distribuir contenidos, usualmente las novedades, de los sitios Web por un canal distinto a la propia página Web.

Gracias a RSS el visitante de una página Web puede suscribirse a sus novedades y recibirlas en su ordenador al instante de ser publicadas, sin necesidad de acceder a la página Web donde se han agregado. RSS está pensado para páginas Web que publican novedades muy a menudo y para usuarios que quieren estar al tanto de tales actualizaciones, sin necesidad de entrar frecuentemente al sitio Web para comprobar nuevas publicaciones.

Para recibir las novedades se tiene que generar una comunicación entre el ordenador del usuario y el servidor donde está la Web. Todo se realiza por medio de un archivo RSS que publica la Web y un lector de RSS instalado en el ordenador del usuario.

4.4.1.1.-Estructura de un documento RSS

Cuando hablamos de un RSS, en realidad estamos tratando un fichero XML con una estructura muy sencilla (como su propio nombre indica, 'Really Simple'). De esta forma, podemos dividir la estructura de un documento RSS en dos apartados:

Channel: Delimitado por la etiqueta <*channel*>, contiene los datos del sitio que produce los datos y la Web propietaria del mismo, con datos de distinta índole como el idioma en el que están los contenidos, la fecha de publicación, los datos del editor, entre otras. Además, dentro del elemento *channel*, se incluyen todos los objetos (*items*) a mostrar.

Items: Cada elemento (delimitado por la etiqueta <*item*>) contiene todos los datos relacionados con la noticia o los datos a mostrar.

A continuación, podemos ver un ejemplo de un documento RSS que podría ser válido para un feed[20] RSS con los últimos mensajes publicados en una red social (por ejemplo, Twitter).

La estructura de dicho ejemplo se compone de un elemento 'channel' con información común a todo el documento RSS, con el título, enlace, descripción e idioma. Junto a estos datos, encontramos información de cada uno de los elementos o 'items' que contiene el ejemplo (cada item tiene su propio título, descripción, fecha de publicación y enlaces).

[20] Feed: Se trata de un medio de redifusión de contenido Web.. Se utiliza para suministrar información actualizada frecuentemente a sus suscriptores.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
    <channel>
        <title> Twitter  /  usuarioX with friends </title>
        <link> http://twitter.com/usuarioX/with_friends </link>
        <description> Twitter updates from usuarioX </description>
        <languaje> es-es </languaje>
        <ttl> 40</ttl>
    <item>
        <title> Uuario X: tweet 1 </title>
        <description> Descripción del tweet</description>
        <pubDate> Tue 27 Jan 2010 14:00 +0000</pubDate>
        <guid>http://twitter.com/usuarioX/1234568</guid>
        <link>http://twitter.com/usuarioX/1234568</link>
    </item>
    <item>
        <title> Uuario X: tweet 2</title>
        <description> Descripción del segundo tweet</description>
        <pubDate> Tue 1 Feb 2010 12:43 +0000</pubDate>
        <guid>http://twitter.com/usuarioX/1234569</guid>
        <link>http://twitter.com/usuarioX/1234569</link>
    </item>
    …
    </channel>…
```

4.4.2.-Atom

Atom [25] se comenzó a diseñar en 2003 como una alternativa al formato RSS, que había nacido en 1999 en el seno de Netscape, el navegador más popular de esa época incipiente de la Web. En 2005 apareció la versión 1.0 de Atom. Las mejoras respecto a RSS se aprecian en cuestiones técnicas; por ejemplo, su código se puede reutilizar dentro de otros que usen el lenguaje de programación XML (la base tanto de Atom como de RSS), una idea pensada para favorecer su extensión. Se trata de un formato de sindicación más moderno que el RSS aunque menos popular.

En el RSS no se puede indicar si la noticia sindicada es un texto plano, un texto con etiquetas HTML o un contenido audiovisual, mientras que en Atom sí. Por otro lado, Atom dispone de más opciones de publicación y se marca con claridad si se envía sólo el titular del texto con un enlace al original, se le añade el primer párrafo o, por el contrario, se manda íntegro todo el contenido. En RSS, en cambio, no hay una forma de comunicar este dato y el usuario no puede saber qué sitios optan por cada una de estas posibilidades. Además, Atom facilita la exportación de los contenidos generados por un medio a otro blog o sitio Web .

A pesar de estas características, RSS predomina incluso cuando se trata de contenidos audiovisuales, como los *podcast*.

Tanto RSS, en sus diferentes formatos, como Atom se pueden leer en la mayoría de lectores de noticias, ya sean de escritorio o de Internet.

4.4.2.1.-Estructura de un documento Atom:

De forma similar a RSS, cuando tenemos un documento en Atom tratamos un fichero XML que tiene unas características determinadas. En este estándar, podemos dividir la estructura en dos apartados:

-Feed: Es muy similar al elemento 'channel' que se define en RSS, y contiene información general sobre el documento, el autor del mismo o la fecha de publicación. La etiqueta *<feed>* se encarga de delimitar este apartado, e incluye atributos importantes como el idioma o el espacio de nombres (Namespace), descrito anteriormente.

-Entry: Se delimita con la etiqueta *<entry>*, y también es análogo a los elementos *item* que encontrábamos en RSS, ya que contienen los datos de cada uno de los elementos que conforman el feed.

A continuación podemos ver el mismo ejemplo que se puso para RSS, pero en este caso para un documento en formato Atom:

```
<?xml version="1.0" encoding="UTF-8"?>
<feed
xml:lang="en-US"  xmlns="http://www.w3.org/2005/Atom"
xmlns:twitter="http://api.twitter.com/">
        <id>tag:search.twitter.com,2005:search/java</id>
        <link type="text/html" rel="alternate"
href="http://search.twitter.com/search?q=java"/>
        <link type="application/atom+xml" rel="self"
href="http://search.twitter.com/search.atom?q=java"/>
        <title>java - Twitter Search</title>
<updated>2009-06-01T12:11:26Z</updated>
        <link type="application/atom+xml"
        rel="next”href="http://search.twitter.com/…=2&q=java">
<entry>
        <id>tag:search.twitter.com,2005:1990561514</id>
        <published>2009-06-01T12:11:26Z</published>
        <link type="text/html" rel="alternate"
        href="http://twitter.com/GailR/statuses/1990561514"/>
        <title>D/L latest upgrade for Google's Chrome  Browser & like it. Faster, esp w
        Java</title>
        <content type="html">D/L latest upgrade for Google's Chrome  Browser &amp;amp;
        like it. Faster, esp w <b>Java</b></content>
        <updated>2009-06-01T12:11:26Z</updated>
        <twitter:source><a href="http://twitter.com/">web</a></twitter:source>
        <twitter:lang>en</twitter:lang>
        <author>
                <name>GailR (Gail R)</name>
                <uri>http://twitter.com/GailR</uri>
        </author>
</entry>
<entry>
… </entry> …
```

Como podemos ver en el ejemplo, y de forma análoga al formato RSS, en primer lugar encontramos información relativa a todo el documento Atom, donde encontramos información global de este feed, como el idioma, fecha de actualización, y diferentes enlaces.

Por otra parte, cada uno de los elementos o ‘items’ del feed aparecen reflejados dentro del documento como ‘entry’. Cada uno de ellos contiene información relativa a cada elemento concreto del feed, con el contenido, fecha de publicación del mismo, enlace, título, autor, etc.

4.4.3.-Conclusión

Una vez estudiadas las alternativas de la sindicación de contenidos, es importante mencionar la importancia que estas pueden tener dentro del proyecto.

En muchas ocasiones es interesante acceder a los distintos *feeds* que las redes sociales tienen a disposición de todos los usuarios de Internet, ya que es posible acceder a datos directamente a través de estos canales de distribución que directamente a través de las API.

Como ejemplo, podemos pensar en Twitter, a través del cual podemos acceder a través de Atom a los últimos tweets de un usuario.

En definitiva, las tecnologías de sindicación de contenido pueden ser de gran utilidad para desarrollar este proyecto, ya que son un medio para obtener datos directamente desde las redes sociales, sin tener que acceder al API de las mismas.

4.5.- JSON

JSON, (*JavaScript Object Notation* [26]), es un formato ligero para el intercambio de datos. Este lenguaje un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX[21]. Una de las ventajas de JSON sobre XML como formato de intercambio de datos en este contexto es que es mucho más sencillo escribir un analizador de JSON. En JavaScript, un documento JSON puede ser analizado de manera trivial usando el procedimiento *eval()*, que ha sido fundamental para su aceptación por parte de la comunidad de desarrolladores AJAX, debido a la presencia de JavaScript en la mayoría de los navegadores Web.

En la práctica, los argumentos a favor de la facilidad de desarrollo de analizadores o del rendimiento de los mismos son poco relevantes, debido a las cuestiones de seguridad que plantea el uso de eval() y el auge del procesamiento nativo de XML incorporado en los navegadores modernos. Por esa razón, JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia (de aquí su uso por Yahoo™, Google™, etc, que atienden a millones de usuarios) cuando la fuente de datos es explícitamente de fiar y donde no es importante el hecho de no disponer de procesamiento XSLT para manipular los datos en el cliente.

Si bien es común ver JSON posicionado frente a XML, también es frecuente el uso de JSON y XML en la misma aplicación.

Esta tecnología será de gran utilidad dentro del proyecto, puesto que algunas redes sociales permiten obtener los datos de su API en este formato. En algunos casos, nos interesa obtener los datos por JSON en lugar de otros formatos de respuesta, ya que presentan los datos de forma más compacta y obtenemos la respuesta con mayor rapidez.

[21] AJAX: acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo Web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

4.6.- Web Services

4.6.1.- ¿Qué son los Web Services?

Los *Web Services* [27] (Servicios Web, en español) son la revolución informática de la nueva generación de aplicaciones que trabajan de forma colaborativa, y en las cuales el software esta distribuido en diferentes servidores.

Cuando definimos un Web Service, hablamos de una interfaz capaz de recibir una petición, activar unos procesos y devolver los resultados. Todo esto, en Internet y a través de protocolos de red (HTTP, FTP, SMTP, etc.). La comunicación entre los diferentes entornos de Web Services se realiza mediante XML.

Para establecer un diálogo coherente entre el WSC (Cliente de Web Services), que envía la petición y recibe la respuesta y el WSS (Servidor de Web Service), el que ejecuta el proceso y envía la respuesta, se utiliza el protocolo SOAP (Simple Object Access Protocol), que es una codificación del servicio basada en XML.

Un Web Service, en vez de obtener peticiones desde un navegador y devolver páginas Web como respuesta, recibe peticiones mediante un mensaje con formato SOAP, terceras aplicaciones realizan la labor pedida y se devuelve el mensaje de respuesta también con formato SOAP.

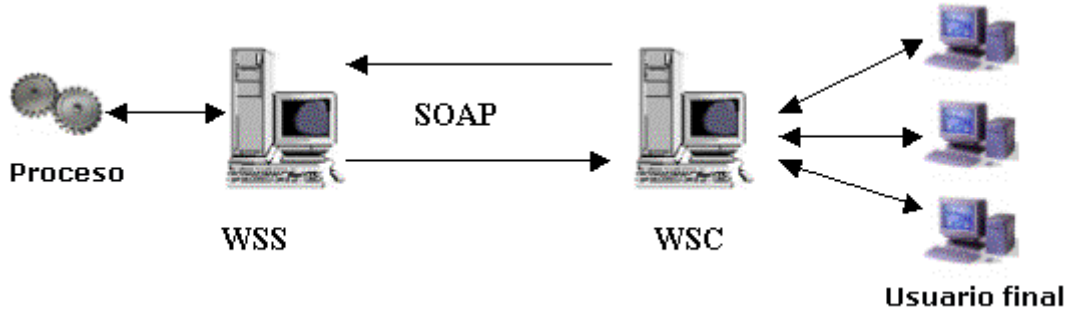

Ilustración 7: Web Services

4.6.2.- ¿Qué utilidad tienen los Web Services?

Un Web Service nos permite tener acceso a información y procesos remotos a través de aplicaciones Web o de escritorio. Esto quiere decir, por ejemplo, que un Web Service puede ser invocado remotamente como una funcionalidad más dentro de una aplicación de escritorio, con algunas ventajas como ser totalmente invisible para el usuario final; dado que se trata de un proceso remoto, el consumo de recursos es absorbido por el Web Service. Otra ventaja que representa el uso de Web Services es que la aplicación puede estar desarrollada en cualquier lenguaje y plataforma.

La finalidad de un Web Service es ofrecer, vender o alquilar un proceso y que este pueda ser invocado por otras aplicaciones sin considerar el lenguaje de programación utilizado.

4.6.3.- SOAP

En el núcleo de los servicios Web se encuentra el protocolo simple de acceso a datos SOAP, que proporciona un mecanismo estándar de empaquetar un mensaje.

SOAP [28] ha recibido gran atención debido a que facilita una comunicación remota denominada RPC (*Remote Procedure Calls*) entre un cliente y un servidor remoto. Existen multitud de protocolos creados para facilitar la comunicación entre aplicaciones, incluyendo RPC[22] de Sun, RMI[23] de Java y ORPC[24] de CORBA.

SOAP ha recibido un gran apoyo por parte de la industria. Es el primer protocolo de su tipo que ha sido aceptado prácticamente por todas las grandes compañías de software a nivel mundial como Microsoft, IBM, Sun, Microsystems, SAP, por mencionar algunas.

Algunas de las ventajas de SOAP son:

- **No está ligado a ningún lenguaje**: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y más potente lenguaje de programación que exista. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como Java, y la plataforma como Microsoft .Net.
- **No se encuentra asociado a ningún protocolo de transporte**: La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que se puede transportar utilizando cualquier protocolo capaz de transmitir texto.
- **Aprovecha los estándares existentes en la industria**: Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado, SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- **Permite la interoperabilidad entre múltiples entornos**: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.

[22] RPC: http://tools.ietf.org/html/rfc5531

[23] RMI: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

[24] RPC: http://www.corba.org/

4.6.3.1.- Estructura de un mensaje SOAP

La estructura básica de un mensaje en SOAP se compone de los siguientes elementos:

- **Envelope:** elemento raíz del formato SOAP. Es la raíz del mensaje, donde se define el documento XML como 'Mensaje SOAP', se describe su contenido y la forma de procesarlo.
- **Header:** elemento opcional, que extiende las funcionalidades básicas de SOAP (seguridad, etc.). Se especifica la información de identificación del contenido.
- **Body:** elemento contenedor de los datos del mensaje. Es obligatoria su presencia.
- **Fault:** si existe error, esta sección contendrá la información sobre el tipo de error. Es opcional

La estructura del mensaje se representaría como se muestra a continuación:

```
<soap:envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingstyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:header>
    ...
  </soap:header>

        <soap:body>
              <soap:fault>

               ...

               </soap:fault>
        ...
        </soap:body>
</soap:envelope>
```

4.7.- REST

Es una técnica de arquitectura software para sistemas hipermedia distribuidos como Web. Se originó en el año 2000, en una tesis doctoral sobre la Web escrita por Roy Fielding[25], uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

REST [29] define un conjunto de principios por los cuales se diseñan Servicios Web haciendo énfasis en los recursos del sistema, incluyendo la forma de acceder al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST se ha convertido en los últimos años en el modelo predominante para el diseño de servicios, logró un impacto tan grande en la Web que prácticamente consiguió desplazar a SOAP y las interfaces basadas en WSDL, al poseer un estilo mucho más simple de usar.

En la actualidad, años después de su presentación, comienzan a aparecer varios entornos REST; se ha convertido además en parte integral de la plataforma Java 6.

Una implementación concreta de un servicio Web REST sigue cuatro principios de diseño fundamentales.

1.- Utiliza los métodos HTTP de manera explícita:

REST hace que los desarrolladores usen los métodos HTTP explícitamente de manera que resulte consistente con la definición del protocolo. Este principio de diseño básico establece una asociación *uno-a-uno* entre las operaciones de crear, leer, actualizar y borrar y los métodos HTTP. De esta forma:

-Se usa POST para crear un recurso en el servidor

-Se usa GET para obtener un recurso

-Se usa PUT para cambiar el estado de un recurso o actualizarlo

-Se usa DELETE para eliminar un recurso

2.- No mantiene estado:

Los servicios Web REST necesitan escalar para poder satisfacer una demanda en constante crecimiento. Se usan clusters de servidores con balanceadores de carga y alta disponibilidad, *proxies* y *gateways*[26], de manera que permita transferir peticiones

[25] Roy Fielding: http://roy.gbiv.com/

[26] Gateway: Se trata de un dispositivo, con frecuencia un ordenador, que permite interconectar redes con protocolos y arquitecturas diferentes a todos los niveles de comunicación. Su propósito es traducir la información del protocolo utilizado en una red al protocolo usado en la red de destino.

de un equipo a otro para disminuir el tiempo total de respuesta de una invocación al servicio Web.

El uso de servidores intermedios para mejorar la escalabilidad hace necesario que los clientes de servicios Web REST envíen peticiones completas e independientes; es decir, se deben enviar peticiones que incluyan todos los datos necesarios para cumplir el pedido, de manera que los componentes en los servidores intermedios puedan redireccionar y gestionar la carga sin mantener el estado localmente entre las peticiones.

La siguiente ilustración muestra un servicio con estado, del cual una aplicación realiza peticiones para la página siguiente en un conjunto de resultados multi-página, asumiendo que el servicio mantiene información sobre la última página que pidió el cliente. En un diseño con estado, el servicio incrementa y almacena en algún lugar una variable *paginaAnterior* para poder responder a las peticiones siguientes.

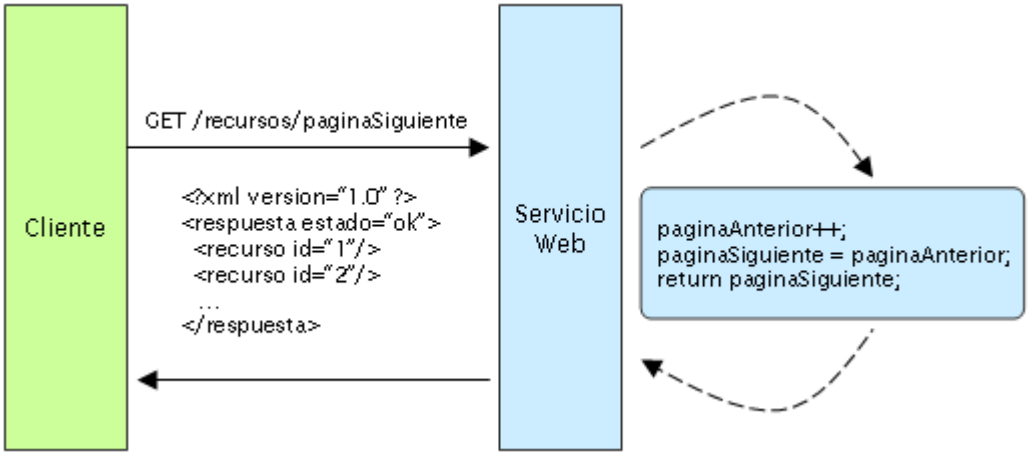

Ilustración 8: Servicio con estado

Por otro lado, los servicios sin estado son mucho más simples de diseñar, escribir y distribuir a través de múltiples servidores. Un servicio sin estado no sólo funciona mejor, sino que además mueve la responsabilidad de mantener el estado al cliente de la aplicación. En un servicio Web REST, el servidor es responsable de generar las respuestas y proveer una interfaz que le permita al cliente mantener el estado de la aplicación por su cuenta.

Por ejemplo, en el mismo ejemplo de una petición de datos en múltiples páginas, el cliente debería incluir el número de página a recuperar en vez de pedir "la siguiente", tal como se muestra en la siguiente figura:

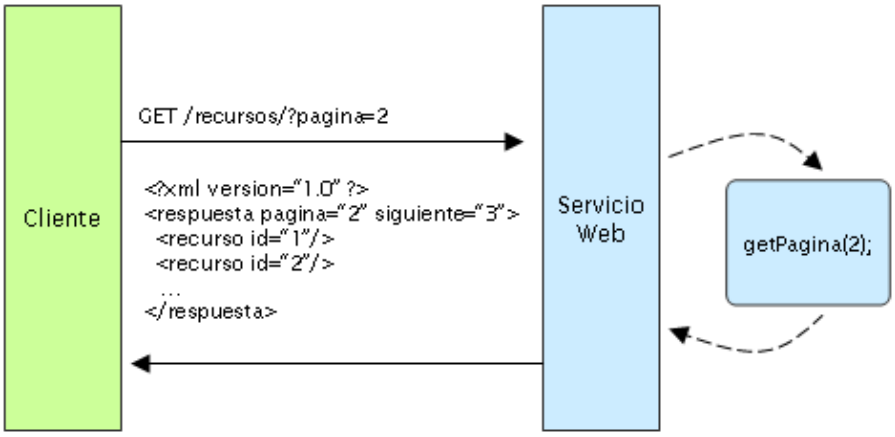

Ilustración 9: Servicio sin estado

3.- Expone URIs con forma de directorios:

Las URI de los servicios Web REST deben ser intuitivas, hasta el punto de que sea fácil adivinarlas. Pensemos en las URI (Uniform Resource Identifiers) como una interfaz que necesita de muy poca o ninguna explicación o referencia para que un desarrollador pueda comprender a lo que apunta, y a los recursos derivados relacionados.

Una forma de lograr este nivel de usabilidad es definir URIs con una estructura al estilo de los directorios. Este tipo de URIs es jerárquica, con una única ruta raíz, y va abriendo *ramas* a través de las sub-rutas para exponer las áreas principales del servicio. De acuerdo a esta definición, una URI no es solamente una cadena de caracteres delimitada por barras, sino más bien un árbol con subordinados y padres organizados como nodos. Por ejemplo, en nuestro caso, se podría definir una estructura de URIs como esta:

http://www.busscall.com/{redsocial}/{tipobusqueda/{resultadospopagina/{keyword}

De esta forma, podemos realizar una búsqueda de un video, por ejemplo, en todas las redes sociales que soporta la aplicación, con 20 resultados por página y la clave prueba, mediante la siguiente URI:

http://www.busscall.com/all/video/20/prueba

Será necesario seguir algunas pautas para configurar este tipo de URIs, como por ejemplo:

-Ocultar la tecnología usada en el servidor que aparecería como extensión de archivos (.jsp, .php, .asp), de manera de poder portar la solución a otra tecnología sin cambiar las URI.

-Mantener la dirección en minúsculas.

-Sustituir los espacios con guiones o guiones bajos.

4.- Transfiere XML, JavaScript Object Notation (JSON), o ambos

La última restricción a la hora de diseñar un servicio Web REST tiene que ver con el formato de los datos que la aplicación y el servicio intercambian en las peticiones/respuestas. Se debe transferir en formato XML, JSON o ambos.

REST es una tecnología de gran utilidad para este proyecto, ya que gracias a ella podremos invocar a nuestra aplicación de una forma realmente intuitiva para el usuario, de esta forma, la accesibilidad de nuestra aplicación mejora en gran medida.

4.8 PHP

Rasmus Lerdorf, miembro del equipo de desarrollo de Apache, creó PHP [30] *(Personal Home Page)* en 1994. Su última versión es la número 5, y será la tecnología que empleemos en este proyecto para procesar la información.

Entre las principales características de esta tecnología, podemos destacar los siguientes aspectos [2]:

- **Embebido en HTML:** Las páginas escritas en PHP son simples páginas en HTML que contienen, además de las etiquetas normales, el código del programa que queremos ejecutar.

- **Multiplataforma:** PHP 5 se ejecuta en multitud de plataformas, sistemas operativos y servidores existentes. Es compatible con los tres servidores líderes del mercado: Apache[27], Microsoft Internet Information Server[28] y Sun Java System Web Server[29].

- **Gran comunidad de apoyo:** PHP 5 se ha escrito bajo el auspicio del Código Abierto. Por lo tanto, existe una comunidad que apoya su desarrollo de manera colaborativa. La ventaja principal es que existen multitud de páginas, listas de correo y foros de debate cuyo tema de conversación es el manejo de esta tecnología. Está en constante crecimiento, ya que los desarrolladores de todo el mundo contribuyen con nuevas funciones que se publican en su sitio Web oficial.

- **Tecnología del lado del servidor:** Los lenguajes del lado del servidor son invisibles para los clientes. Las páginas que utilicen *scripts* de este tipo contienen el código entre etiquetas *parecidas* a las de HTML, pero éstas desaparecen cuando el cliente recibe la página. Es decir, el desarrollador incluye el código PHP dentro de páginas HTML. Es en definitiva, transparente al usuario final.

[27] Apache: http://www.apache.org/

[28] Microsoft Internet Information Server: http://www.iis.net/

[29] Sun Java System Web Server: http://www.sun.com/software/products/web_srvr/index.xml

En la siguiente ilustración [3], podemos comprobar el funcionamiento de las páginas PHP.

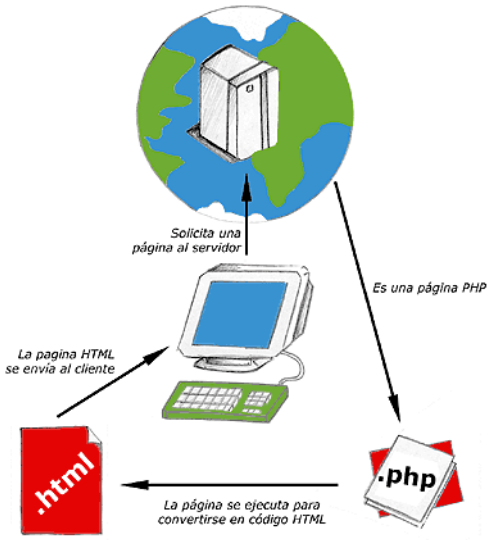

Ilustración 10: Funcionamiento de las páginas PHP

PHP será el lenguaje con el cual será implementado este proyecto, utilizando como complemento otros elementos como XML, Xpath y el resto de tecnologías descritas a lo largo de este capítulo.

4.9.- Parsers XML: DOM, SAX y SimpleXML

Dado que XML es un lenguaje utilizado ampliamente en el desarrollo de la World Wide Web, existen herramientas y estándares de programación para leer y analizar documentos XML. Las herramientas o programas que analizan el lenguaje XML y comprueban si el documento es válido se denominan analizadores sintácticos o "parsers".

Un parser XML [31] es un módulo, biblioteca o programa que se ocupa de transformar un archivo de texto en una representación interna. En el caso de XML, como el formato siempre es el mismo, no necesitamos crear un *parser* cada vez que hacemos un programa, sino que existen un gran número de analizadores sintácticos disponibles que pueden comprobar que un documento XML cumpla con una determinada gramática.

Dada la importancia del desarrollo del lenguaje XML en los últimos años, PHP 5 incorpora tres métodos de acceso a este tipo de archivos: SAX, DOM y SimpleXML. En los sucesivos apartados veremos una descripción de cada uno de estos analizadores.

4.9.1.- SAX

Simple API for XML (SAX) funciona en base a eventos y métodos asociados. A medida que el analizador va leyendo el documento XML y encuentra los componentes del documento (elementos, atributos, valores, etc) o detecta errores, genera un evento e invoca las funciones asociadas por el programador.

Algunas ventajas de SAX sobre DOM es que generalmente SAX utiliza menos memoria que DOM. Adicionalmente, a menudo el procesamiento de documentos por SAX es más rápido por su naturaleza de manejo de eventos. SAX también permite leer a partir de flujos.

Por el contrario, una desventaja de SAX es que algunas clases de validaciones XML requieren acceso al documento completo desde el primer instante.

4.9.2.- DOM

Son las siglas de *Document Object Model* [32]. Se trata de un API que define un completo camino para crear, definir y analizar la sintáctica de archivos XML. DOM es una recomendación del consorcio *World Wide Web* (W3C). La idea básica consiste en que todos los archivos XML pueden verse como un conjunto de nodos que forman parte de un árbol. Empezando desde el elemento raíz, del que todos los elementos nacen como hijos, cualquier programa debería ser capaz de crear una estructura lógica del documento.

La API puede utilizarse para leer archivos en memoria, modificarlos y volver a escribir el archivo con los nuevos datos. DOM está escrito en metodología orientada a objetos, y todos los nodos son instancias de diferentes objetos.

La verdadera potencia de DOM reside en el poder de gestionar los archivos para añadir, modificar o eliminar nodos. La mayoría de los algoritmos que realizan alguna de estas operaciones se basan en la búsqueda de los nodos para, posteriormente, añadir hijos nuevos o eliminar los encontrados.

4.9.3.- SimpleXML

Probablemente, el punto más fuerte de PHP 4 fue la incorporación de herramientas para el soporte de XML. PHP 5 le da una vuelta más a la tuerca implementando nuevas herramientas de lectura, elaboración y modificación de archivos XML con una nueva API llamada SimpleXML [33].

Si XML es un lenguaje bien construido y legible por personas y ordenadores, los programas para manipular estos archivos deberían ser también sencillos de utilizar. SimpleXML nace con esta premisa permitiendo leer un fichero completo con una sola instrucción para, inmediatamente después, poder leer los datos como un conjunto de variables PHP.

El conjunto de funciones que pertenece a la API SimpleXML es nuevo en PHP 5. Mantiene una absoluta flexibilidad a favor de la simplicidad y bajo nivel de memoria usado. SimpleXML utiliza el menor número de líneas de código para leer o escribir datos en un fichero XML.

4.10 Conclusión

XML se ha convertido en un estándar para el intercambio de información en Internet. Numerosas aplicaciones están construidas con algún tipo de tecnología que implementa XML, como XML-RPC, RSS o SOAP.

En este capítulo, hemos descrito cómo funcionan todas las tecnologías que serán necesarias para llevar a cabo este proyecto, de forma que haremos uso de ellas para contactar con las APIs que las redes sociales ponen a nuestra disposición. Posteriormente trataremos estos datos mediante los diferentes manejadores que hemos expuesto anteriormente y generaremos un documento XML que aglutine toda la información recogida, para posteriormente presentarla en una aplicación Web.

5.- Análisis

5.1.- Finalidad del Sistema

Es difícil darse cuenta cómo cambia nuestra vida a no ser que paremos un momento y nos situemos diez o incluso cinco años atrás. Probablemente en los últimos años las redes sociales han impactado en nuestras vidas más que cualquier otra cosa. A principios de esta década se nos hubiera hecho difícil visualizar el panorama global que se ha desarrollado durante los últimos años a través de herramientas como Facebook, Twitter o Flickr, las cuales han cambiado la forma de interactuar entre nosotros y con el mundo.

El número de redes sociales que podemos encontrar en la red aumenta cada día, existiendo actualmente comunidades que integran redes sociales relacionadas con cualquier ámbito de la vida, como música (MySpace[30], LastFM[31]), fotografía (Flickr[32]), pequeños mensajes de texto (Twitter[33]), perfiles personales (Tuenti[34], Facebook[35]) y otras temáticas.

El objetivo que se persigue con el desarrollo de este proyecto es crear una herramienta que permita integrar un conjunto de redes sociales con gran cantidad de usuarios actualmente en Internet, como Twitter, Flickr o Facebook, entre otras. De esta forma podremos realizar búsquedas en todas ellas con un solo clic de ratón, integrando así los contenidos presentes en cada una de ellas.

Para mejorar el resultado de nuestra aplicación, obtendremos la salida en uno de los estándares más utilizados en Internet como es XML, que otorga portabilidad al resultado dado que permite ser procesado en diferentes plataformas (móviles, webs, PDA's, etc.).

Podemos encontrar algunas herramientas en Internet con un enfoque orientado al que tiene este proyecto, como FriendFeed[36]o Tumblr[37], las cuales permiten que con una cuenta podamos acceder a los datos de nuestras otras cuentas en Twitter, Facebook y otras redes sociales. Estas aplicaciones integran los datos de un mismo usuario en una sola página o aplicación Web; por nuestra parte, el objetivo es desarrollar una nueva utilidad que permita buscar datos en un conjunto de redes sociales y obtener una salida estándar que nos permita leerla en distintos dispositivos, con las posibilidades que esto proporciona al usuario.

---

[30] Myspace: http://es.myspace.com/

[31] LastFM: http://www.lastfm.es/

[32] Flickr: http://www.flickr.com/

[33] Twitter: http://www.twitter.com/

[34] Tuenti: http://www.tuenti.com

[35] Facebook: http://www.facebook.com

[36] FriendFeed: http://www.friendfeed.com/

[37] Tumblr: http://www.tumblr.com/

5.2.- Requisitos

Para desarrollar este proyecto vamos a seguir el ciclo de desarrollo que comprende tareas como el análisis, diseño, implementación y evaluación.

Dentro de la fase de análisis, podemos encontrar la especificación de requisitos que describe las funcionalidades que debe tener la aplicación. Siguiendo la metodología descrita en el capítulo 3, hemos diferenciado entre requisitos funcionales y no funcionales.

Los requisitos funcionales, definen el comportamiento interno de la aplicación: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que muestran cómo los casos de uso serán llevados a la práctica. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación.

Los requisitos no funcionales, especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales.

En resumen, los requisitos funcionales establecen los comportamientos del sistema y los no funcionales se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar.

5.2.1.- Requisitos Funcionales

5.2.1.1- Busscall

Identificador	RF001
Nombre	Intercambio de datos.
Descripción	Las diferentes clases y elementos que componen la aplicación deben ser capaces de intercambiar datos entre ellas y proceder al tratamiento de los mismos cuando sea necesario.
Justificación	Para un correcto funcionamiento de la aplicación, es necesario que los distintos componentes del sistema estén integrados y entre sí.

Identificador	RF002
Nombre	Uso de API's de las redes sociales.
Descripción	La aplicación se comunicará y obtendrá datos a través de la API de cada una de las redes sociales en las cuales se efectúe la búsqueda.
Justificación	Es necesario obtener los datos por esta vía.

Identificador	RF003
Nombre	Obtención de un documento XML.
Descripción	Los resultados obtenidos de las distintas redes sociales se integrarán en un documento XML.
Justificación	Obtenemos una salida en un formato estándar que nos permite tratarlo en multitud de plataformas y presentar los resultados en cualquier formato deseado.

Identificador	RF004
Nombre	Documento XML bien formado.
Descripción	El documento XML de salida será un documento bien formado que se validará contra una estructura estándar definida a través de un fichero XML Schema
Justificación	Es necesario obtener una salida válida y bien formada.

Identificador	RF005
Nombre	Uso de ficheros de configuración.
Descripción	La aplicación se configurará de forma automática mediante la lectura de un fichero de configuración que contendrá las distintas redes sociales que se encuentran activas y funcionales.
Justificación	Desarrollar un sistema escalable y fácilmente configurable.

Identificador	RF006
Nombre	Integración de nuevas redes sociales.
Descripción	La aplicación se podrá ampliar mediante la integración de nuevas redes sociales y la actualización del fichero de configuración. Para ello será necesario implementar las funciones mínimas requeridas.
Justificación	Desarrollar un sistema escalable.

5.2.1.2- Aplicación ejemplo

Identificador	RF007
Nombre	Buscador.
Descripción	La aplicación dispondrá de un campo de texto para indicar las palabras clave o *keywords* con las cuales debe llevarse a cabo la búsqueda en las diferentes redes sociales.
Justificación	Es necesario que el usuario indique qué es lo que desea encontrar.

Identificador	RF008
Nombre	Selección de objetivos.
Descripción	Junto al campo de texto, se mostrarán 'checkbox' (selectores) que permitan al usuario seleccionar las redes sociales en las que desea llevar a cabo cada una de las búsquedas. A su vez, existirá un selector para realizar la búsqueda en todas las redes sociales disponibles.
Justificación	Con el objetivo de optimizar el rendimiento, el usuario puede marcar un número determinado de redes sociales en las que esté interesado en recopilar datos, ahorrando así el procesamiento de redes que no son de su interés.

Identificador	RF009
Nombre	Almacenamiento del documento de salida.
Descripción	La aplicación almacenará un fichero físico dentro de la carpeta resultados, que contendrá el documento XML que el módulo de búsqueda proporciona como salida.
Justificación	Eliminar el hecho de tener que intercambiar contínuamente el mismo documento entre diferentes páginas, aumentando la rapidez y rendimiento del sistema.

Identificador	RF010
Nombre	Retorno al home
Descripción	Se podrá volver a la página inicial desde cualquiera de las páginas que componen la aplicación, mediante un logo identificativos situado siempre en la misma posición.
Justificación	Permite utilizar la herramienta de forma sencilla.

Identificador	RF011
Nombre	Presentación de resultados
Descripción	Cuando se realice la búsqueda, la aplicación redirigirá automáticamente a la aplicación ejemplo que tratará el fichero XML almacenado, para presentar los resultados tratando dicho fichero.
Justificación	Permite utilizar la herramienta de forma sencilla.

Identificador	RF012
Nombre	Cambio en el tamaño de texto
Descripción	La aplicación permitirá modificar el tamaño del texto de los resultados que se presenten.
Justificación	Permite que usuarios con problemas de carácter visual puedan tener acceso a la aplicación.

Identificador	RF013
Nombre	Filtrado de resultados
Descripción	Los resultados podrán ser filtrados en función de varios criterios, como red social, fecha, descripción, título y autor.
Justificación	Facilitar al usuario final el manejo de los resultados obtenidos

Identificador	RF014
Nombre	Acceso directo a la información.
Descripción	Se podrá acceder a los distintos contenidos obtenidos sin necesidad de salir de la aplicación, pudiendo así reproducir un vídeo, ver una imagen, o leer el texto encontrado haciendo click en el enlace habilitado para ello.
Justificación	Mejora de la accesibilidad.

Identificador	RF015
Nombre	Acceso directo al contenido.
Descripción	Se podrá acceder directamente al contenido dentro de la red social haciendo click en un enlace habilitado, que abrirá una nueva ventana o pestaña dentro del navegador.
Justificación	Mejora de la accesibilidad.

Identificador	RF015
Nombre	Nube de etiquetas.
Descripción	En la parte izquierda de la aplicación aparecerá una nube de etiquetas con los conceptos que describen las características de los resultados encontrados. Haciendo click en cada etiqueta, se podrá acceder a los resultados que se ajusten a la misma
Justificación	Mejora del manejo de los resultados obtenidos.

Identificador	RF015
Nombre	Realizar nuevas búsquedas y obtener más páginas
Descripción	Se podrá realizar una nueva búsqueda así como avanzar entre las diferentes páginas de resultados.
Justificación	Mejora de la accesibilidad.

5.2.2.- Requisitos No Funcionales

5.2.1.1- Busscall

Identificador	RNF001
Nombre	Tecnología a emplear
Descripción	Para el desarrollo de la aplicación se hará uso de las distintas tecnologías explicadas en apartados anteriores, como XML, JavaScript, HTML, XPath, REST etc.
Justificación	Estas tecnologías nos proporcionan la capacidad para llevar a cabo el proyecto.

Identificador	RNF002
Nombre	Uso de URL's limpias.
Descripción	Se podrá acceder a la aplicación a través de direcciones limpias y sencillas de recordar, mediante la utilización de la tecnología REST.
Justificación	Mejora de la accesibilidad.

5.2.1.2- Aplicación ejemplo

Identificador	RNF003
Nombre	Compatibilidad con varios navegadores
Descripción	La aplicación debe visualizarse y ejecutarse de forma correcta en los principales navegadores del mercado (Internet Explorer, Mozilla Firefox y Ópera).
Justificación	Permitir el acceso correcto a la gran mayoría de perfiles de usuarios.

Identificador	RNF004
Nombre	Presentación adecuada
Descripción	La aplicación utilizara una combinación de colores adecuada para garantizar la usabilidad y correcta visualización de los contenidos
Justificación	Hacer legible el contenido de la aplicación.

Identificador	RNF005
Nombre	Mínimo de caracteres para una búsqueda
Descripción	Es necesario que el elemento a buscar tenga una longitud mínima de 3 caracteres.
Justificación	Maximizar el rendimiento y evitar resultados demasiado grandes e imprecisos.

Identificador	RNF006
Nombre	Notificación de errores
Descripción	Si se produce un error en los datos de entrada porque no cumplen los requisitos (es decir que al menos haya una red social seleccionada, y se cumpla la longitud mínima) se mostrará un error.
Justificación	Es necesario que el usuario esté informado cuando se produce un error durante la ejecución de la aplicación.

Identificador	RNF007
Nombre	Títulos identificativos
Descripción	La aplicación contará con títulos de página identificativos respecto a su contenido.
Justificación	Facilitar la identificación de entre varias ventanas/pestañas.

6.- Diseño

Una vez hemos estudiado la finalidad del sistema y su viabilidad, procedemos a realizar el diseño para mejorar la implementación.

Se busca que Busscall se caracterice por ser una aplicación con alta portabilidad y escalabilidad, además de destacar por ofrecer los datos utilizando un formato estándar, con las ventajas que ello conlleva.

La siguiente ilustración muestra de forma gráfica cómo se ha diseñado el sistema.

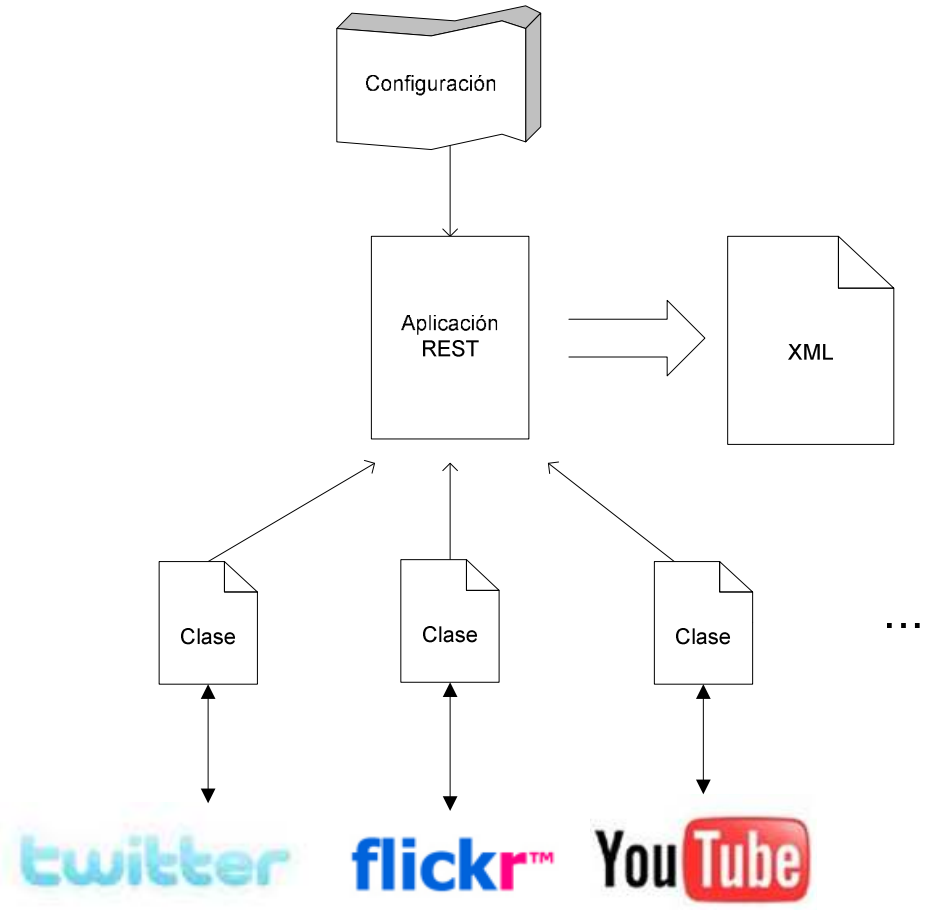

Ilustración 11: Diagrama de funcionamiento de Busscall

Como se aprecia en la ilustración, el sistema accede a cada una de las API de las redes sociales. Para ello, anteriormente consulta un fichero de configuración que contiene toda la información relativa a las redes sociales activas, y las clases que manejan cada una de ellas. Una vez que estas API devuelven su información, ésta es procesada y se genera un documento XML.

De forma complementaria, se ha desarrollado una aplicación de ejemplo que maneja el documento XML y presenta los datos de forma gráfica, permitiendo filtrarlos en función de varios criterios y acceder a la información directamente.

Una de las bases bajo las que se ideó este proyecto es su escalabilidad. Busscall permite de forma sencilla añadir nuevas redes sociales, haciendo cada vez más potente su funcionalidad permitiendo obtener más resultados. Para ello, se ha diseñado un modelo que describe las características para todas las redes sociales que puede soportar el sistema, de forma que añadir una nueva sea una tarea sencilla. Para ello, cada clase que se integra en el sistema, corresponderla con una red social diferente y debe cumplir varios requisitos.

Cada una de las clases da soporte a una red social, y se encuentra dentro de la carpeta 'redes', en un archivo cuyo título corresponde con el nombre de cada red social. Todas las clases se han desarrollado de forma análoga siguiendo un mismo patrón, de forma que tienen como titulo el mismo nombre que la red con la que trabaja, y a su vez incluyen un conjunto común de funciones que tienen el mismo título en todas las clases, el cual también será igual para nuevas redes que se integruen en el sistema. Las funciones mínimas que toda red social debe incluir son:

- *Busqueda*: Se encargara de recuperar los datos de la red social correspondiente
- *InsertarResultadosXML*: Inserta un conjunto de resultados dentro del XML de salida, sin incluir la información relativa a los totales.
- *InsertarResultadosXMLconTotal*: Incluye junto con los resultados individuales, los datos totales en relación a dicha red y los resultados insertados.
- *InsertarResultado* (*): Inserta dentro del XML de salida un resultado individual.

* Esta función no es obligatoria pero si es muy recomendable incluirla, para favorecer la claridad del código y una estructura común en todas las clases.

Casos de Uso

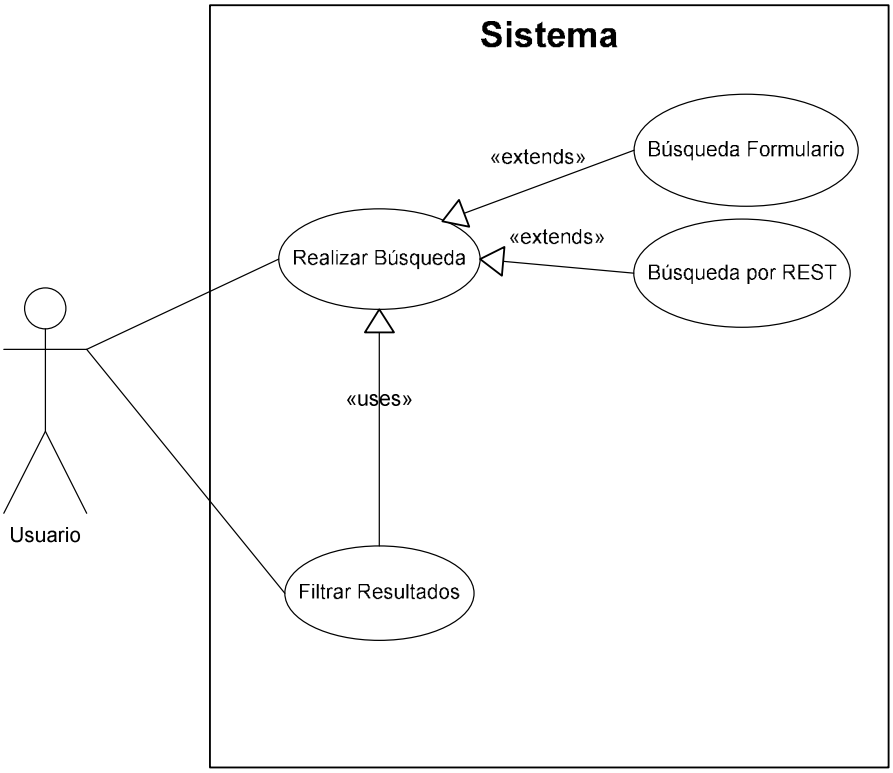

Ilustración 12: Diagrama de casos de uso

En el diagrama de casos de uso superior, podemos ver como el usuario de esta aplicación tiene dos posibles acciones (íntimamente relacionadas).

Un usuario puede simplemente realizar una búsqueda y obtener los resultados ordenados por defecto.

También puede filtrar los resultados que la aplicación muestra para tenerlos ordenados en función del criterio que considere oportuno, ya sea por red social, fecha, titulo, descripción o autor.

Por otra parte, la aplicación tiene dos posibilidades de realizar la búsqueda. Una de ellas es accediendo directamente a la aplicación ejemplo, mientras que por otra parte, se puede invocar a la aplicación REST para obtener el documento XML de salida.

Diagramas de actividad

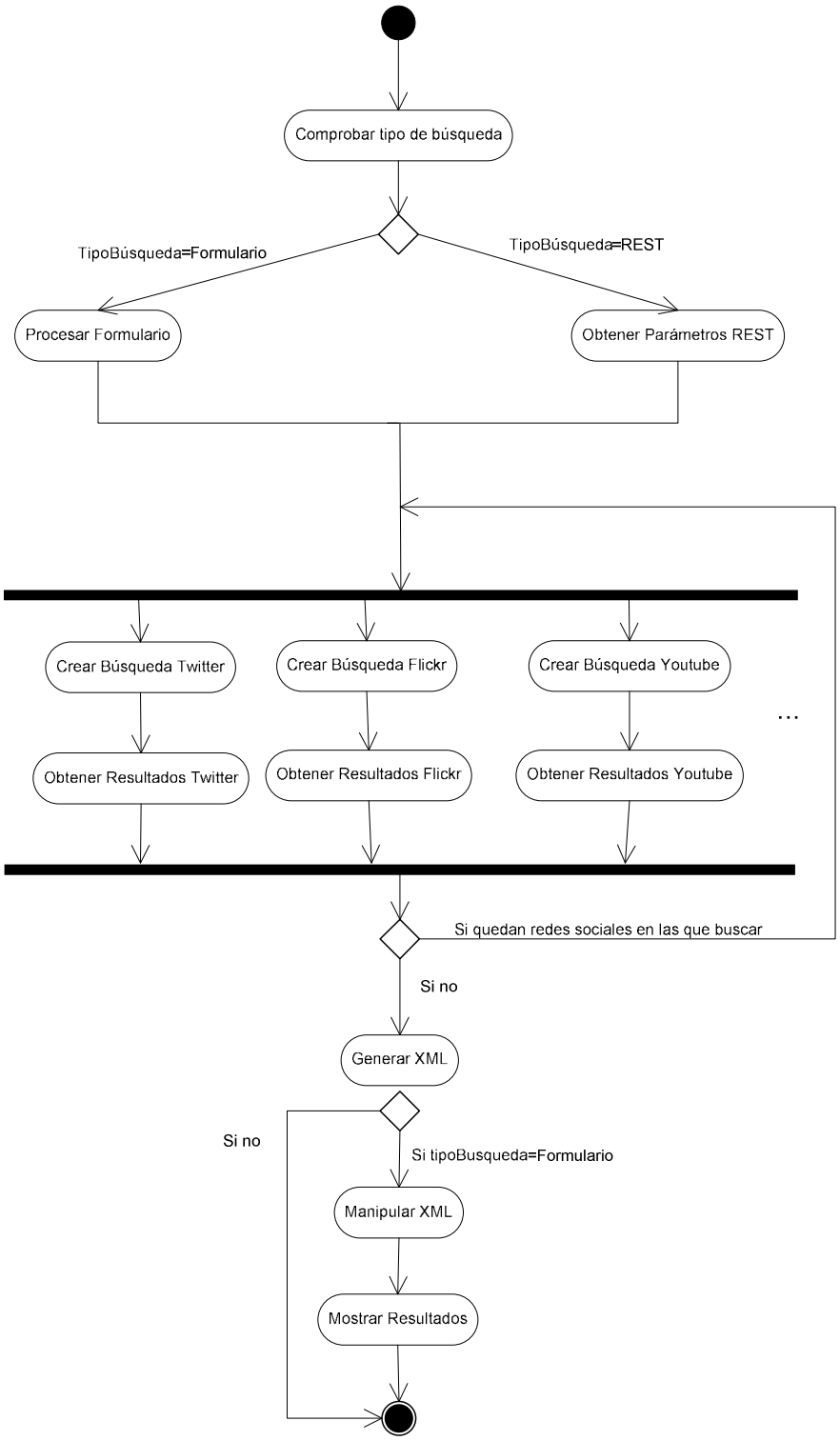

Ilustración 13: Diagrama de actividad del módulo de búsqueda

La figura superior, representa el diagrama de actividad de la aplicación principal. En función de cómo se acceda a la aplicación (por REST o por formulario), se realizan algunos pasos de forma diferente, como se explica a continuación.

Cuando se accede por REST a la aplicación, se obtienen los parámetros de la búsqueda directamente de la dirección introducida en el navegador. Si por el contrario, se accede a la aplicación por formulario, se leerán los datos introducidos para generar una dirección REST con ellos e invocar a la aplicación. Seguidamente, el sistema debe ir obteniendo los datos de las diferentes redes sociales que se hayan solicitado, y cuando termine generará el documento XML de salida. Si hemos accedido por REST, aquí finaliza la aplicación, mientras que si se accede por formulario (a través de nuestra aplicación ejemplo), se almacena una copia del documento XML en el servidor y se llama a la aplicación de ejemplo para que muestre los resultados en un entorno gráfico.

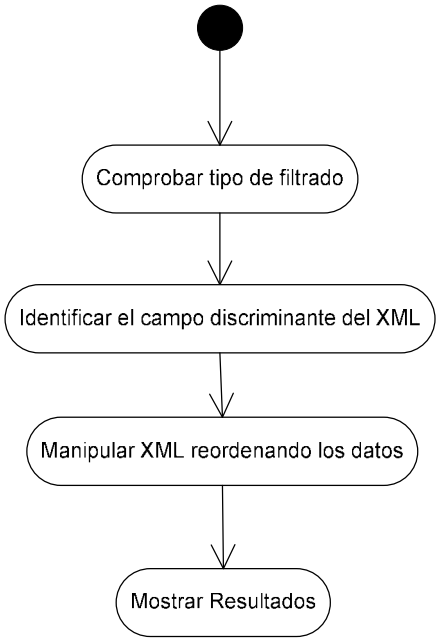

Ilustración 14: Diagrama de actividad del filtrado de resultados

El diagrama de actividad superior nos muestra los pasos que se siguen cuando se solicita filtrar los datos de acuerdo a un criterio concreto.

En primer lugar se obtiene el criterio de filtrado y posteriormente se obtienen los resultados a mostrar. Se ordenan los mismos teniendo en cuenta el criterio de ordenación, y por último se muestran en la aplicación Web.

Diagrama de secuencia

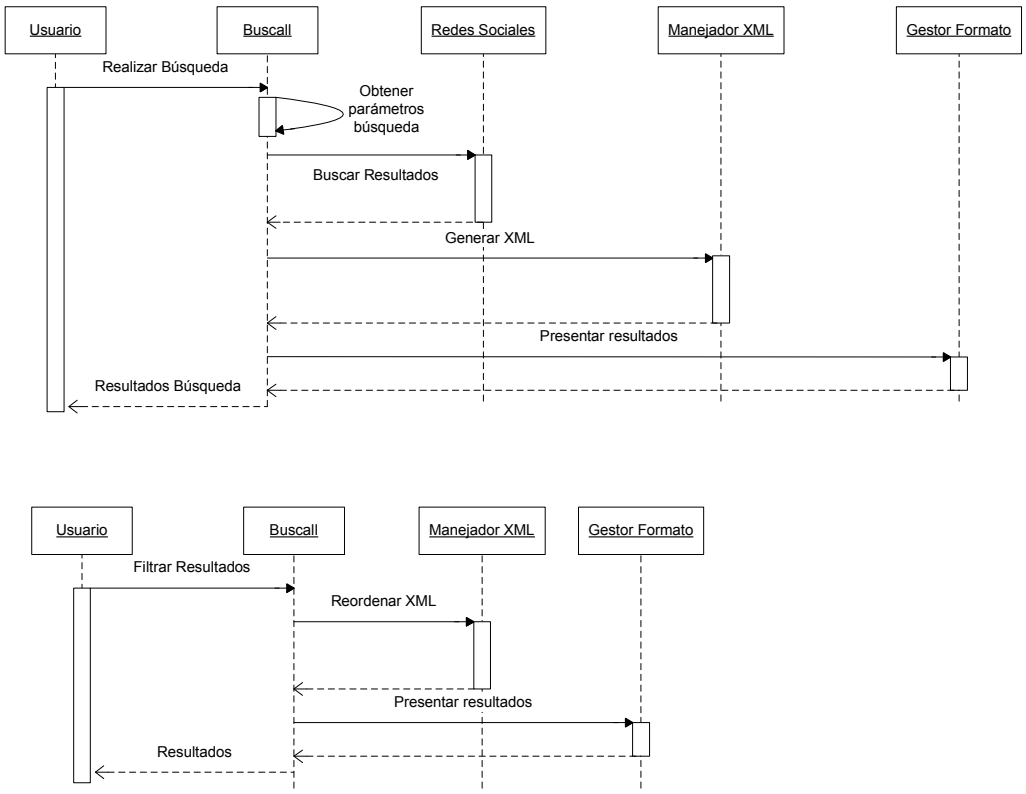

Ilustración 15: Diagrama de secuencia de la aplicación

De igual forma que en los diagramas anteriores, podemos ver en este caso los diagramas de secuencia de la aplicación. El primero de ellos nos indica los pasos que se siguen en el momento de realizar la búsqueda. En el segundo caso, observamos como se lleva a cabo el filtrado de los resultados.

El usuario solicita la operación y posteriormente Busscall se comunica con los diferentes elementos que componen el sistema, como las redes sociales, los manejadores XML que utiliza PHP y la aplicación de ejemplo que se encarga de dar un formato gráfico a los resultados obtenidos.

7.- Implementación

En esta sección se detalla exhaustivamente cómo se ha implementado este proyecto. Por ello, este capítulo, se divide en dos partes. La primera de ellas tratará exclusivamente la aplicación que se comunica directamente con las distintas redes sociales y obtiene los datos (Busscall), mientras que por otra parte, encontraremos la descripción detallada de la aplicación que se encarga de procesar los datos y mostrarlos al usuario (nuestra aplicación de ejemplo).

Como se ha indicado anteriormente, el proyecto se divide en dos grandes aplicaciones que se comunican entre sí y permiten tener una encapsulación mayor, ya que por una parte se consulta y se obtienen los resultados, y por otra se van tratando y mostrando los mismos.

La más importante de las dos es la encargada de conectarse directamente con el API de cada una de las redes sociales. Esta parte del proyecto es, de hecho, la base del mismo, ya que sin ella nuestra aplicación de ejemplo, la que se encarga de mostrar los datos, ordenarlos, filtrarlos, etc. No tendría sentido.

Por otra parte, se ha desarrollado una segunda aplicación, que nos permite ver la utilidad que tiene este proyecto. Nuestra aplicación Web, bien podría haber sido una aplicación para Android, para iPhone o cualquier otro dispositivo, por lo que la salida estandar que obtenemos es verdaderamente importante.

7.1.- Busscall

Esta aplicación constituye la base principal del proyecto, ya que el resto de elementos del mismo, solicitan la información que esta aplicación concreta devuelve.

Para acceder a Busscall, hemos utilizado tecnología REST. El motivo de escoger esta tecnología es la comodidad y facilidad que transmite al usuario. De esta forma, si queremos realizar una búsqueda con el objetivo de obtener como respuesta el documento XML que nuestra aplicación genera, podremos acceder de forma sencilla desde el navegador. Para ello, el prototipo de dirección a utilizar es la siguiente:

http://localhost/busscall/ buscar / **A** / **B** / **C** / **D** / **E**

Los elementos destacados son los verdaderamente relevantes, ya que el principio de la dirección, dependerá de donde se encuentre alojada nuestra aplicación.

Cada una de las partes de la dirección tiene un significado:

-**A**: Se indica el tipo de búsqueda a realizar. En caso de ampliar el proyecto, podríamos realizar no sólo búsquedas por redes sociales, sino también por autor, fecha tipo de contenido…

-**B**: Se indica el objetivo de la búsqueda, separando varias redes sociales con un '-'. En caso de requerir una búsqueda en todas las redes sociales, esta parte de la dirección se indica como 'all'.

-**C**: Se introducen el conjunto de palabras clave por las que realizar la búsqueda.

-**D**: Se establece el número de resultados por página deseado. Busscall está limitado a 200 resultados por página como máximo

-**E**: Se establece el número de página de la búsqueda. Es decir, si por ejemplo hemos realizado una búsqueda de 35 elementos, la primera página contendrá los resultados del 1 al 35 , la página 2 del 36 al 70, y así sucesivamente.

Si se desea acceder a la aplicación por REST, es necesario rellenar todos los campos de la dirección correctamente, ya que si no obtendremos un error puesto que existen todos los datos son necesarios para una correcta ejecución de la misma.

Por otra parte, como se ha comentado anteriormente, una de las principales características que tiene el sistema es que proporciona su salida en un formato estándar como es XML. En las siguientes páginas se encuentra toda la información relativa a la estructura que presenta el documento XML que genera la aplicación.

Taxonomía del documento XML

BUSQUEDA
- RESULTADOS
  - RESULTADO
    - Red Social
    - Autor
    - Titulo
    - Descripción
    - Información
    - Fecha
    - Hora
    - Contenido
    - Extensión *
    - Thumb
    - Origen
    - Etiquetas *
      - Etiqueta
      - …
    - Categoría *
    - Extra *
    - Ítem
      - …
  - TOTAL
  - Red
  - Numero de Resultados
  - Página
  - Total de Resultados *
  - Total de Páginas *

La estructura del documento XML está descrita y validada por el documento XSD que aparece reflejado en el Anexo I.

Como podemos ver, el documento XML se estructura en varios niveles. En primer lugar podemos ver como toda la información se encuentra contenida dentro del elemento BÚSQUEDA. Dentro de este nivel, tendremos el conjunto de elementos RESULTADOS, el cual almacena el conjunto de resultados de cada red social (cada ocurrencia de RESULTADO) y algunos datos globales sobre dicha red: (TOTAL), con datos como el nombre de la red, el número de resultados que contiene dicho XML, el total de resultados disponibles para dicha red, etc.

En cada documento XML, sólo podremos tener un elemento BUSQUEDA. Este elemento puede contener entre uno y N elementos RESULTADOS. Cada conjunto de resultados se compone de N elementos RESULTADO y un único elemento TOTAL y representa los resultados encontrados en una red social concreta.

A continuación aparece una relación de los distintos campos del elemento RESULTADO, su descripción y número de repeticiones de cada uno de ellos en cada ocurrencia.

Resultado		
Campo	Descripción	Repeticiones
Red Social	Nombre de la red social donde se encuentra el resultado concreto.	1
Autor	*Nickname* del usuario que publicó el resultado obtenido.	1
Título	Título descriptivo asociado al resultado.	1
Descripción	Contiene una descripción acerca del contenido del resultado.	1
Información	Enlace directo a la información (video o imagen) o texto que contiene el resultado propiamente dicho.	1
Fecha	Fecha en la que se publicó el resultado.	1
Hora	Hora en la que se publicó el resultado.	1
Contenido	Tipo de contenido que tiene el resultado (TEXTO | IMAGEN | VIDEO).	1 a N
Extension	Extensión del archivo que contiene la información	0 o 1
Thumb	Enlace a una imagen descriptiva del resultado. Por ejemplo el avatar de un usuario de Twitter o una imagen en miniatura de una fotografía en Flickr.	1
Etiquetas	Almacena un conjunto de ocurrencias del campo "etiqueta".	0 o 1
Etiqueta	Contiene una palabra clave relacionada con el resultado.	1 a N
Categoría	Indica la categoría en la que se enmarca el resultado	1
Extra	Contiene un conjunto de ocurrencias del campo "Item"	0 o 1
Item	Cada uno de estos elementos contiene información adicional que va variando en función del resultado. Puede contener datos tan variopintos como "Fotógrafo", "Fecha de captura", "Valoración"…	1 a N
Origen	Contiene un enlace que accede directamente al resultado, accediendo a su red social.	0 o 1

Tabla 1: Estructura documento XML (resultado)

Cada conjunto de resultados tiene un elemento TOTAL que contiene la información global relativa a ellos.

Total	
Campo	Descripción
Red	Nombre de la red social que agrupa un conjunto de resultados.
Numresult	Número de resultados de esta red social que contiene el documento XML.
Página	Número de página de los resultados del documento XML.
TotResult	Total de resultados encontrados para esta red social.
TotPáginas	Total de páginas de resultados para esta red social

Tabla 2: Estructura documento XML (red social)

Busscall es una aplicación escalable, y para desarrollar este punto hemos utilizado ficheros de configuración que, una vez editados, modificarán las distintas redes sociales desde las cuales se obtendrá información. El fichero que contiene estos datos lo hemos denominado *'busquedaRedes.conf'*, que contiene una entrada por cada una de las redes sociales soportadas por Busscall y que almacena datos como el nombre de la red social o el número máximo de resultados que esta permite obtener con una llamada del sistema. En estos ficheros se permite introducir comentarios que serán ignorados por la aplicación y que permiten presentarlo de forma clara. Estos comentarios comienzan por punto y coma (;).

El hecho de implementar este proyecto de forma escalable es un aspecto que permite manejarlo y extenderlo de forma muy rápida sin tener que modificar el código principal de la aplicación. A su vez, podemos utilizarlo de forma inversa desactivando el funcionamiento de cierta red social, que por ejemplo se encuentra fuera de servicio o no existe actualmente. A continuación se detalla como se debe proceder para poder añadir funcionalidades al sistema, incluyendo una red social nueva y activándola.

Una vez se ha creado una clase, la cual debe contener las funciones que se detallan en el capítulo de diseño, es necesario activarla actualizando la configuración. Para ello, se añade una nueva entrada en el fichero con el siguiente formato:

[NOMBRE RED SOCIAL]
Nombre = Nombre red social
Fichero = redsocial.php
Icono = resocial-icon.png
Max = 50

Una vez explicados los detalles relativos al documento XML y lso ficheros de configuración, se detallará el funcionamiento de la aplicación en ejecución. Busscall obtiene los parámetros de la búsqueda de la dirección del navegador, y comienza a trabajar.

En primer lugar, se procede a comprobar que los parámetros obtenidos son correctos, de forma que no se supere el número máximo de resultados permitidos, o eliminar posibles redes sociales repetidas en la entrada, entre otros.

Cuando se solicita la búsqueda en más de una red social, se divide el número de resultados solicitados entre el número de redes en las que se debe buscar de forma que obtenemos el numero de resultados a obtener de cada red social. Cuando este valor no es entero, se redondea hacia el entero más cercano. De esta forma, no se da prioridad a ninguna red social y siempre se devuelve el número de resultados más próximo al solicitado. Supongamos que solicitamos 25 resultados a obtener en 3 redes sociales. Para cada red social, se deberían obtener 8,33 resultados aproximadamente, por lo que el valor queda redondeado a 8 y en total se mostraran 24 resultados. Lo mismo sucede con 50 resultados para 3 redes sociales, en cuyo caso, el valor se redondea al alza (16,66 pasa a ser 17 resultados por red) y se obtendrán 51 resultados por página.

Posteriormente lee los ficheros de configuración, recogiendo los datos relativos a las redes sociales de las cuales se quiere obtener información, con datos como la ubicación del fichero que trabaja directamente con dicha red social, o el número máximo de resultados que el API de dicha red nos permite obtener en cada momento, lo cual es importante a la hora de mostrar los resultados, ya que por ejemplo, Twitter tiene una limitación de 100 'tweets', Flickr sólo permite obtener hasta 100 resultados por cada llamada, y Youtube 50. De esta forma, este es un dato importante para poder unificar todos los resultados solicitados en una misma salida.

Una vez obtenidos los datos de configuración, vamos realizando un conjunto de acciones para cada una de las redes sociales solicitadas. En primer lugar, se importa el fichero que corresponde a la red social que se trata en ese momento, cuya ruta se conoce gracias a los ficheros de configuración. Posteriormente, existen dos vías para llevar a cabo las búsquedas:

- Opción 1 → El número de resultados solicitados es menor que el máximo permitido por la red social.

  -En este caso, se realiza una búsqueda simple ya que podemos obtener todos los resultados con una sola llamada.

- Opción 2 → Cuando se solicita un número de resultados mayor que el máximo permitido por la red social.

  -En esta situación, tenemos que dividir la búsqueda en varias, de forma que podamos obtener el número de resultados solicitados, teniendo en cuenta las limitaciones que tiene el API.

Cada vez que se realiza una búsqueda, los datos recogidos, se van insertando dentro de lo que posteriormente será nuestra salida XML. Para ello, cada red social tiene una función determinada, que se encarga de separar e insertar cada uno de los campos que tendrá la salida.

Cuando se han insertado todos los datos de una red social, se introduce en el XML los datos globales de dicha red, como el número de resultados insertados, el nombre de la red social, los resultados totales que se pueden encontrar, la página y el total de páginas, en función de la información que cada una proporcione.

Se itera de la misma forma para cada una de las redes sociales en las cuales se solicita la búsqueda. Una vez se ha generado completo el XML, se muestra en el navegador de forma tabulada y agradable para el usuario.

7.2.- Aplicación Ejemplo

Una vez que nuestra aplicación principal está desarrollada, hemos decidido crear esta aplicación para demostrar la utilidad que tiene este proyecto. El hecho de obtener una salida XML permite que cualquier dispositivo pueda tener una aplicación específica, incluso con diferentes funcionalidades.

En este caso hemos desarrollado una aplicación con una interfaz gráfica que permite visualizar todos los resultados obtenidos en la búsqueda y a su vez ordenarlos en función de distintos criterios, como la red social en la que se encuentra, fecha, título, o autor. A su vez y directamente en la aplicación podemos acceder al contenido, visualizando el contenido sin tener que salir de la aplicación.

Busscall, presenta una primera pantalla en la cual se solicita al usuario insertar el conjunto de palabras clave que le interesa buscar. A su vez, se debe seleccionar las redes sociales en las cuales se quiere realizar la búsqueda. Esta pantalla muestra de forma dinámica las redes sociales disponibles para realizar una búsqueda, accediendo al fichero de configuración y mostrando solo aquellas redes que están activas.

Una vez se ha rellenado el formulario, la aplicación abre una nueva búsqueda y construye la dirección REST a través de la cual llamará a Busscall. De esta forma, obtiene los datos como si de una llamada a un API se tratara. Una vez ha recogido dichos datos, almacena un fichero XML dentro de la carpeta *resultados*. El hecho de tener el fichero almacenado, permite una mayor velocidad, a la hora de filtrar los resultados, evitando tener que repetir la búsqueda. Cada documento XML, se identifica con un código de cinco caracteres generado antes de su creación. Este código será utilizado por la aplicación para identificar el documento dentro de la carpeta de resultados, lo que evita tener que enviar continuamente toda la información relativa a los resultados entre las distintas páginas.

Una vez se ha almacenado el fichero XML, la aplicación abre una conexión CURL y llama a la página encargada de mostrar los resultados. Por defecto, la primera vez que se realiza una búsqueda se ordenan los resultados por red social de forma ascendente. Para ello, la aplicación va recorriendo todos los resultados y los ordena en función del criterio solicitado.

En la parte izquierda de la aplicación, se podrá realizar una nueva búsqueda sin tener que regresar a la página de inicio de la aplicación. Además se podrá solicitar el filtrado de resultados en función de los criterios fijados (red social, título, descripción, autor y fecha). Por último encontraremos una nube de etiquetas que contiene un conjunto de palabras, o conceptos relacionados con la búsqueda realizada. Pinchando en cada una de las etiquetas, obtendremos sólo aquellos resultados que concuerdan con la misma. Cada vez que se solicita filtrar los resultados, la aplicación vuelve a abrir una conexión CURL con la página que muestra los resultados, indicándole el orden que debe seguir y el código del documento XML que contiene los resultados.

Nuestra aplicación de ejemplo se estructura en 3 partes. En la cabecera, podemos encontrar el logo y la posibilidad de cambiar el tamaño del texto. En la parte inferior a la cabecera, tenemos dos partes. En el lado izquierdo, se encuentran las distintas formas de filtrado, así como la nube de etiquetas y el formulario para realizar una nueva búsqueda. En la parte derecha se encuentra el contenido, con cada uno de los resultados encontrados y las posibles acciones que cada uno de ellos permite.

Toda la estructura de la aplicación de ejemplo se ha llevado a cabo a través de una hoja de estilo CSS cuyo objetivo es presentar los datos de forma clara y limpia para el usuario.

La aplicación, a su vez, también tiene en cuenta algunas cuestiones relacionadas con la usabilidad, de forma que se puede visualizar en tres tamaños distintos, estando por defecto activado el tamaño mediano o normal, pudiendo este ser reducido o aumentado.

En función del tipo de resultado que se vaya mostrando, la aplicación permitirá realizar unas acciones u otras. A través de JavaScript, si tenemos un video, podremos visualizarlo directamente en la aplicación. Lo mismo sucede con las imágenes, de forma que podremos ver cada una de ellas en un tamaño más grande que el que se muestra en un principio en la aplicación.

Se ha utilizado la librería *colorbox* de JavaScript, disponible en Internet y con licencia libre, para mostrar cada uno de los resultados dentro de la propia aplicación de ejemplo. De forma sencilla, podemos ampliar la imagen o reproducir el video que se ha encontrado en alguna de las redes sociales.

Por último, en la parte inferior, podremos solicitar una nueva página. La nueva búsqueda se realizará siguiendo los criterios de la búsqueda actual, es decir, con el mismo número de resultados por página y las mismas redes sociales como objetivo.

8.- Caso de uso

En este capítulo se muestra un ejemplo de uso de la aplicación y se comparan los resultados con lo que obtenemos en las distintas redes sociales.

En este caso, el concepto a buscar va a ser 'UC3M'. A continuación, podemos ver varios ejemplos de llamadas a la aplicación REST.

Para buscar la primera página de resultados de UC3M en una red social, como por ejemplo Twitter, con 20 resultados por página, introducimos la siguiente dirección en nuestro navegador:

http://localhost/busscall/buscar/redes/ **twitter** / **UC3M** /20/1

Seguidamente, vemos la dirección a introducir si queremos obtener la tercera página

http://localhost/busscall/buscar/redes/twitter/UC3M/20/ **3**

También podemos cambiar el número de resultados

http://localhost/busscall/buscar/redes/twitter/UC3M/ **75** /1

Y las redes sociales en las que realizar la búsqueda. Vamos a buscar en Youtube, FlickR y Twitter a la vez:

http://localhost/busscall/buscar/redes/ **twitter-flickr-youtube** /UC3M/20/1

Por otra parte, si deseamos cambiar la palabra clave a buscar:

http://localhost/busscall/buscar/redes/twitter-flickr-youtube/ **Universidad** /20/1

Para este ejemplo, vamos a realizar una búsqueda en las tres redes sociales y solicitaremos 15 resultados por página y la primera página de resultados.

La dirección a introducir en el navegador será:

http://localhost/busscall/buscar/redes/twitter-flickr-youtube/UC3M/15/1

En la siguiente figura, vemos una captura de pantalla del aspecto que tiene el navegador cuando se ha realizado la operación. En la mayoría de navegadores actuales, podremos ver y manejar de forma sencilla el XML generado.

El XML resultante se puede consultar en el Anexo II de este documento. Los valores que se introducen en el documento XML de salida, son codificados antes de añadirlos al mismo, de forma que podemos encontrar alguna variación al consultar el documento en caracteres que pueden presentar algún conflicto de codificación, como acentos, letras como la 'ñ' y similares.

```
−<busqueda xsi:noNamespaceSchemaLocation="xml/estructuraResultados.xsd">
  −<parametros>
      <objetivo>twitter-flickr-youtube</objetivo>
      <key>UC3M</key>
      <rpp>15</rpp>
      <pag>1</pag>
    </parametros>
  −<resultados>
    −<resultado>
        <redsocial>Twitter</redsocial>
        <autor>jberlana</autor>
      −<titulo>
          Tweet del usuario 'jberlana', 2010-05-17 a las 12:11:12.
        </titulo>
      −<descripcion>
          Poco ambiente de estudio en la biblioteca de la #uc3m.
        </descripcion>
      −<informacion>
          Poco ambiente de estudio en la biblioteca de la #uc3m.
        </informacion>
        <fecha>2010-05-17</fecha>
        <hora>12:11:12</hora>
        <contenido>TEXTO</contenido>
      −<thumb>
          http://a1.twimg.com/profile_images/389745954/Captura_de_pantalla_2009-08-30_a_las_20.18.34_normal.png
        </thumb>
      −<etiquetas>
          <etiqueta>uc3m</etiqueta>
        </etiquetas>
      </resultado>
```


Ilustración 16: XML de salida

A continuación vamos a proceder a utilizar la aplicación de ejemplo que se ha desarrollado en el proyecto para realizar la misma búsqueda que se ha hecho directamente con REST. A su vez, comprobaremos que los valores que nos devuelve cada una de las redes sociales en su propia página Web, coinciden con los que Busscall nos está proporcionando.

En primer lugar, debemos acceder a la aplicación. Para ello, accedemos desde nuestro navegador a la siguiente dirección:

http://localhost/busscall

Una vez dentro de la aplicación, rellenamos el campo del formulario con la palabra clave a buscar y a su vez, marcamos las redes sociales de las cuales deseamos recuperar resultados y el número de resultados por página que se requiere.

Ilustración 17: Formulario de búsqueda

Pinchamos en el botón de buscar y esperamos a que la aplicación termine de hacer su trabajo. En la siguiente ilustración, vemos el aspecto que presenta nuestra aplicación de ejemplo una vez que ha obtenido los resultados.

Ilustración 18 : Interfaz de la aplicación ejemplo

Cuando la aplicación muestra un resultado, en función del tipo del mismo (imagen, video o texto) permite realizar diferentes acciones. Cuando tenemos una imagen, podemos hacer clic en el enlace de 'Ver Imagen' y automáticamente podremos obtener la imagen ampliada, como se puede ver en la siguiente imagen.

Ilustración 19: Visualización de contenido (imagen)

Para salir de la ventana flotante que muestra la imagen, sólo es necesario pinchar en el fondo negro, o en el botón creado a tal efecto justo en la parte inferior derecha de la imagen. A su vez, podemos navegar entre las diferentes imágenes que obtenidas como resultados, mediante las dos flechas que aparecen a en la parte inferior izquierda.

De forma análoga al caso de las imágenes, cuando se recupera un video podemos reproducirlo directamente en la aplicación.

Ilustración 20: Visualización de contenido (video)

Por otra parte, cuando el resultado a mostrar sólo contiene texto, no se facilita ninguna posibilidad de este tipo ya que toda la información ya aparece reflejada.

Llegados a este punto, se va a comparar que los resultados que Busscall está proporcionando corresponden en realidad con lo que se obtendría si se realiza una búsqueda desde la página Web de cada una de las redes sociales.

Ilustración 21: Resultados obtenidos en Busscall

Ilustración 22: Resultados obtenidos en twitter.com

Como podemos comprobar, los resultados obtenidos, aunque están presentados de distinta forma, son exactamente los mismos.

Ilustración 23: Resultados obtenidos en Busscall

Ilustración 24: Resultados obtenidos en flickr.com

De igual forma que con Twitter, podemos comprobar como los resultados son exactamente iguales. Busscall, proporciona, a su vez, determinada información adicional que en Flickr sólo es accesible si se pincha en cada una de las fotografías por separado.

Cabe destacar que Busscall recupera siempre los datos en función de la fecha, obteniendo siempre los últimos. Flickr, por defecto realiza la búsqueda siguiendo el criterio de relevancia, por lo que se debe tener en cuenta a la hora de comparar los resultados.

Ilustración 25: Resultados obtenidos en Busscall

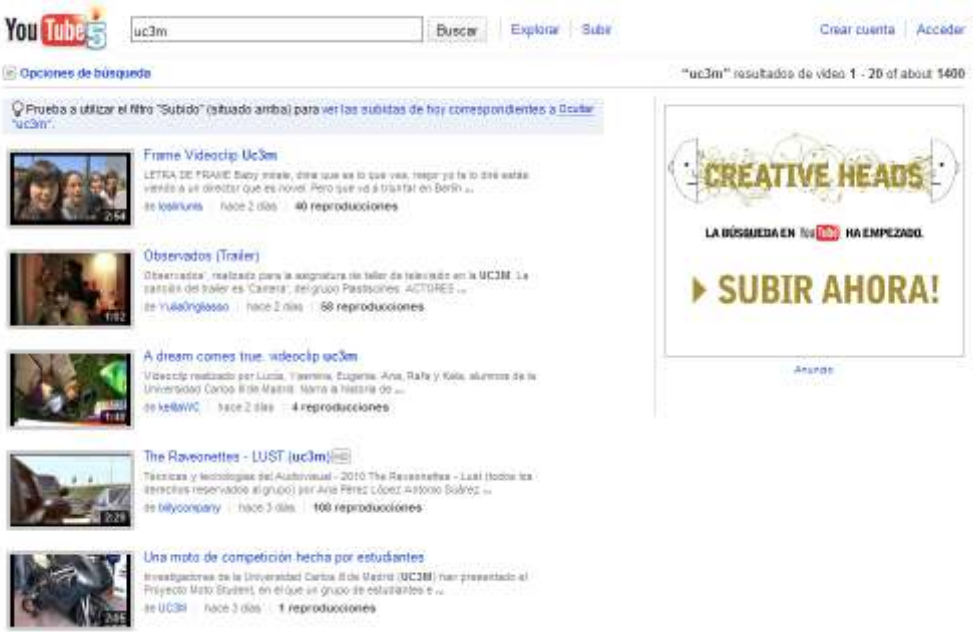

Ilustración 26: Resultados obtenidos en youtube.com

De la misma forma que en los casos anteriores, los resultados son los mismos en Busscall y Youtube. Es importante tener en cuenta que la búsqueda en Youtube se ordene por el criterio de la fecha de subida.

8.1.- Casos de error

En esta sección se detallarán algunas situaciones que pueden provocar un posible error dentro de la aplicación. Varios casos concretos pueden proponerse como futuras mejoras del sistema.

- Búsqueda menor a tres caracteres: Busscall requiere que la búsqueda mínima a realizar tenga una longitud mayor o igual tres caracteres. En caso contrario, se desplegará un mensaje que invita al usuario a repetir la búsqueda indicando porqué no se ha podido llevar a cabo.

- Una o varias API no se encuentran disponibles: En el improbable caso de que el API de alguna de las redes sociales esté fuera de servicio, busscall tendrá un error crítico para el cual no proporciona soporte. De esta forma se propone como una posible mejora futura del sistema.

- Saturación de peticiones: En algunos casos, las API pueden estar saturadas de peticiones, bien por que no soportan el conjunto de datos que requieren todos los usuarios o porque la cuenta de un desarrollador concreto ha quedado deshabilitada durante unos minutos al superar el tráfico máximo permitido. De forma análoga al caso anterior, no se ha desarrollado soporte para estas situaciones ya que durante el desarrollo de la aplicación no se han producido este tipo de incidentes por lo que se propone como una mejora futura del sistema.

- Exceso de resultados por página: Busscall permite realizar búsquedas con un máximo de 200 resultados por página. En caso de solicitar un número mayor de resultados, se desplegará un mensaje indicando al usuario la razón por la cual no se ha llevado a cabo la búsqueda. No obstante, para la aplicación de ejemplo el número de resultados máximo que se permite realizar está limitado por defecto a 200, por lo que este caso de error solo se puede presentar cuando se realiza una búsqueda por REST directamente a través del módulo de búsqueda.

9.- Validación

En este capítulo vamos a explicar las características que hacen de Busscall un sistema escalable. La implementación que se ha llevado a cabo intenta hacer más sencilla esta tarea, y para ello se utilizan ficheros de configuración.

Si queremos añadir la capacidad de realizar búsquedas para una nueva red social, debemos crear una clase con una determinada estructura. Para ello y como se explicó en el capítulo de diseño, las funciones mínimas que toda red social debe incluir son:

- *Busqueda*: Se encarga de recuperar los datos de la red social correspondiente
- *InsertarResultadosXML*: Inserta un conjunto de resultados dentro del XML de salida, sin incluir el la información relativa a los totales.
- *InsertarResultadosXMLconTotal*: Incluye junto con los resultados individuales, los datos totales en relación a dicha red y los resultados insertados.
- *InsertarResultado* (*): Inserta dentro del XML de salida un resultado individual.

* Esta función no es obligatoria pero si es muy recomendable incluirla, para favorecer la claridad del código y una estructura común en todas las clases.

Para este ejemplo, vamos a utilizar la red social Photobucket, que aloja tanto imágenes como vídeos. Una vez hemos consultado la documentación de su API, sabemos que una vez que nos conectemos, podremos realizar búsquedas de contenido, ya sea imágenes o videos.

Photobucket nació en 2003 y permite alojar tanto imágenes como vídeos. Fue adquirida por la productora cinematográfica Fox en Mayo de 2007. El objetivo de esta red social es similar a Flickr, permitiendo compartir videos y fotografías con otros usuarios, agregar contactos y otras funcionalidades comunes como los comentarios para la valoración del contenido. Está integrada con otras redes sociales como Facebook o Twitter.

Photobucket dispone de un API realmente completa, y para utilizarla es necesario solicitar una 'key'. Para ello es necesario registrarse como usuario de Photobucket y obtener una clave de desarrollador.

Disponer de una clave de desarrollador, nos permite acceder a los servicios que proporciona el API de Photobucket. En este momento, es preciso consultar la documentación que el API nos proporciona. En este caso concreto, encontramos una parte dedicada a la búsqueda de contenido a través del API[38].

En esta documentación, encontramos todas las instrucciones para realizar una llamada al API de Photobucket con el propósito de obtener contenido. Disponemos de varios parámetros a configurar a nuestro gusto.

[38] http://pic.pbsrc.com/dev_help/WebHelpPublic/Content/Examples/Searching%20Media.htm

Además, al hacer la llamada al API es necesario que nos identifiquemos con nuestra clave de desarrollador. Para ello tenemos que adjuntar los datos que se nos proporcionan al solicitar la clave de desarrollador dentro de la URL que obtendrá los resultados.

De forma genérica, la URL de la que podemos obtener datos sigue el formato que aparece a continuación, especificando nuestros datos de desarrollador, la búsqueda a realizar y los elementos por página deseados (datos marcados en negro). Previamente, es necesario autenticarse en el API como desarrollador, donde se nos proporcionarán algunos parámetros para la sesión actual, como el timestamp o la firma (datos desatacados en rojo).

http://api.photobucket.com/search/**BUSQUEDA**?oauth_consumer_key=**CLAVE_PUBLICA**&oauth_nonce=**CLAVE_PRIVADA**&oauth_signature=**FIRMA**&oauth_signature_method=HMAC-SHA1&oauth_timestamp=**TIMESTAMP**3&oauth_version=1.0&perpage=**RPPPAGINA**

Cuando se realiza la llamada al API, obtenemos una respuesta (puede verse en el Anexo III). Con los datos que recibimos en la respuesta (que se recibe en formato XML), podremos programar los métodos que se encargarán de ampliar Busscall para esta red. Debemos seleccionar qué campo de la respuesta que nos proporciona el API corresponden con los campos establecidos para el XML de salida que crea Busscall, y qué campos recibimos como respuesta nos son de utilidad para adjuntarlos como información extra, así como aquellos que queramos desechar.

Dentro del documento XML que se recibe como respuesta, están los datos con los que tendremos que rellenar nuestros métodos. Los métodos descritos al comienzo este capítulo, son esenciales para el buen funcionamiento de la clase, por lo que se va a describir de forma más detallada qué deberían hacer en este caso.

- *Busqueda*: Para recuperar los datos, en primer lugar es necesario construir la dirección concreta de la búsqueda a realizar. Esta función se encargará de ello y posteriormente recuperará los datos de la dirección construida, convirtiendolos a un formato manejable por PHP.
- *InsertarResultadosXML*: A través de este procedimiento, se deben insertar en el el documento XML parte de los resultados encontrados en la búsqueda. Es posible que sean necesarias varias llamadas al API, porque el número de resultados a recuperar supere el máximo permitido por Photobucket. Este método se utiliza cuando los datos a insertar no son los últimos. Para ello, se va recuperando cada resultado concreto y se va invocando a la función encargada de insertar los resultados, sin insertar las estadísticas totales de esta red social en la búsqueda
- *InsertarResultadosXMLconTotal*: Se trata de una función parecida a la anterior, con la diferencia que en este caso los datos que se integran en el fichero de salida son los últimos de los encontrados para esta red social. De esta forma, se hace necesario insertar las estadísticas de la red. También se utilizará cuando el número de resultados a recuperar sea menor que el máximo permitido, y pueda hacerse toda la búsqueda con una sola llamada.

- *InsertarResultado* (*): Este procedimiento, puede no existir, ya que su contenido se puede integrar en las dos funciones anteriores. Por comodidad y claridad del código se recomienda incluirlo. Este método debe recibir un resultado e ir recuperando e identificando cada uno de los campos a incluir en el XML de salida. Por ejemplo en photobucket, al recibir un resultado, obtendrá el usuario que subió el contenido navegando por el resultado y llegando a la ruta 'media [username]'. Otro dato sencillo de obtener es la URL que contiene la imagen, contenido en la ruta 'media->url'. De forma análoga con el resto de campos, se deben ir recopilando para finalmente insertarlos en el documento de salida. Esta función se invocará una vez por cada resultado a insertar.

Una vez que las funciones indicadas anteriormente estén implementadas correctamente, se procederá a activar la red social en Busscall. Para ello debemos editar el fichero de configuración, de forma que debemos añadir al final del mismo los datos relativos a esta red social, por ejemplo:

[PHOTOBUCKET]

Nombre = Photobucket

Fichero = photobucket.php

Icono = photobucket-icon.png

Max = 50

De esta forma, debemos llamar a la clase que contiene las funciones que hemos creado 'photobucket.php' e insertarla dentro de la carpeta 'redes' que contiene el resto de clases ya implementadas. A su vez, añadiremos la imagen con el icono en tamaño pequeño de la red social dentro de la carpeta 'images/icons' de nuestra aplicación ejemplo, así como un logotipo de tamaño mayor dentro de la carpeta 'images/logos'.

Para detectar las redes que están disponibles para realizar una búsqueda, en el momento de generar la página principal, la aplicación sigue los siguientes pasos:

1.- Recupera el fichero de configuración.

2.- De forma iterativa recupera las redes sociales que aparecen en el mismo.

3.- Con los parámetros que incluye cada red social, genera una nueva ocurrencia en la que poder seleccionar dicha red en el formulario de entrada.

De esta forma, y una vez actualicemos, Busscall, tendrá habilitada la capacidad de buscar en esta nueva red social, y automáticamente aparecerá para ser seleccionada, como se refleja en la siguiente imagen.

Ilustración 27: Formulario de búsqueda

Desde este momento, y si no hay ningún fallo de programación, la aplicación será totalmente funcional con la nueva red agregada. Esta característica hace de Busscall una aplicación con un potencial importante dado el gran número de redes sociales que actualmente están implantando su API para permitir acceder a sus contenidos desde aplicaciones externas.

10.- Conclusiones

Una vez finalizado el desarrollo y la validación de este proyecto se puede observar el aporte que se proporciona para desarrolladores y usuarios finales, permitiendo acceder a una aplicación escalable e interoperable utilizando una estructura de datos uniforme y flexible, a través de la cual se pueden realizar búsquedas personalizadas en diferentes redes sociales.

Es necesario evaluar si los objetivos con los que nació este proyecto se han llevado a cabo durante el tiempo de desarrollo. Se ha conseguido desarrollar una aplicación que accede a las API de las diferentes redes sociales, y procesa la información obtenida para generar una salida que permite al usuario obtener una búsqueda personalizada y a su vez proporciona una salida en que puede ser reutilizada por otros sistemas o desarrolladores.

De igual forma, los datos recuperados siempre corresponden con los más actuales disponibles, y la aplicación de ejemplo desarrollada, nos proporciona una idea para futuras implementaciones, haciendo uso de la salida estándar proporcionada por el módulo de búsqueda.

Como se puede leer a lo largo de este documento, existen alternativas similares a este proyecto, pero el propósito de este desarrollo consiste en aportar algo más, de forma que desarrolladores puedan hacer uso de la aplicación para crear nuevas aplicaciones para un amplio abanico de dispositivos y permitiendo una ampliación del las funcionalidades.

Trabajos Futuros

A su vez, existen diferentes mejoras futuras que pueden proponerse, como:

- Modificar la estructura de configuración definida actualmente por otro sistema más robusto y seguro, utilizando por ejemplo un Sistema Gestor de Base de Datos.
- Explotar más las funcionalidades que las API proporcionan para permitir publicar contenido, permitiendo al usuario publicar contenido en las redes sociales, almacenar sus últimas búsquedas, etc.
- Proporcionar la posibilidad de obtener la salida en otro formato estándar, como JSON.
- Implementar la aplicación utilizando las tecnologías más actuales, por ejemplo, combinando HTML5 y Web Sockets permitiendo una actualización en tiempo real de los resultados.
- Aumentar los tipos de contenidos soportados, permitiendo recuperar también sonido, presentaciones en powerpoint, archivos PDF, etc.

Opiniones Personales

Por último, y de forma personal, este proyecto me ha proporcionado la oportunidad de familiarizarme con tecnologías muy utilizadas actualmente, como PHP, XML o Xpath. Considero que tecnologías como XML o Json tienen un gran futuro por delante, una vez he comprobado que su uso es cada vez mayor en el desarrollo de aplicaciones Web.

Las redes sociales, a su vez, representan un entorno en constante crecimiento y que será de gran importancia para las empresas en los próximos años, constituyendo un nuevo medio de comunicación y captación de clientes.

Por tanto, considero que dirigir mi proyecto de fin de carrera al ámbito de las redes sociales, utilizando las tecnologías mencionadas ha sido una decisión muy acertada.

11.-Tabla de Acrónimos

Acrónimo	**Significado**
XML	eXtensible Markup Language
S/GML	Lenguaje de marcado Standard / Generalized Markup Language
DTD	Documento de definición de tipos (Document Type Definitions)
XSL/T	Extensible Stylesheet Language / Transformations
JSON	JavaScript Object Notation
RSS	Formato de sindicalización Rich Site Summary o Really Simple Syndication
Atom	Formato de sindicalizacion
REST	Representational State Transfer
SAX	Simple API for XML
DOM	Document Object Model
API	Application Programming Interface

Tabla 3: Tabla de acrónimos

12.-Anexos

Anexo I:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="busqueda">
 <xs:complexType>
 <xs:sequence>
 <xs:element name="parametros" maxOccurs="1">
 <xs:complexType>
  <xs:sequence>
          <xs:element name="objetivo" type="xs:string"/>
          <xs:element name="key" type="xs:string"/>
        <xs:element name="rpp" type="xs:positiveInteger"/>
        <xs:element name="pag" type="xs:positiveInteger"/>
</xs:sequence>
  </xs:complexType>
  </xs:element>
<xs:element name="resultados" maxOccurs="unbounded">
 <xs:complexType>
<xs:sequence>
    <xs:element name="resultado" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
         <xs:element name="redsocial" type="xs:string"/>
         <xs:element name="autor" type="xs:string"/>
         <xs:element name="titulo" type="xs:string"/>
         <xs:element name="descripcion" type="xs:string"/>
         <xs:element name="informacion" type="xs:string"/>
         <xs:element name="fecha" type="xs:date"/>
         <xs:element name="hora" type="xs:time"/>
         <xs:element name="contenido" type="xs:string" maxOccurs="unbounded"/>
         <xs:element name="extension" type="xs:string" minOccurs="0"/>
         <xs:element name="thumb" type="xs:string"/>
         <xs:element name="etiquetas" maxOccurs="1" minOccurs="0">
         <xs:complexType>
         <xs:sequence>
         <xs:element name="etiqueta" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
                  </xs:sequence>
         </xs:complexType>
         </xs:element>
         <xs:element name="categoria" type="xs:string" maxOccurs="unbounded" minOccurs="0"/>
         <xs:element name="extra" maxOccurs="1" minOccurs="0">
         <xs:complexType>
                  <xs:sequence>
                  <xs:element name="item" type="xs:string" maxOccurs="unbounded" minOccurs="1"
maxOccurs="unbounded">
                  <xs:complexType mixed="true">
                           <xs:attribute name="tipo" type="xs:string" use="optional" />
                  </xs:complexType>
                  </xs:element>
                  </xs:sequence>
         </xs:complexType>
         </xs:element>
```

```
          </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="total">
      <xs:complexType>
        <xs:sequence>
                                <xs:element name="red" type="xs:string"/>
          <xs:element name="numresult" type="xs:positiveInteger"/>
          <xs:element name="pagina" type="xs:positiveInteger"/>
          <xs:element name="totresult" minOccurs="0" type="xs:positiveInteger"/>
          <xs:element name="totpaginas" minOccurs="0" type="xs:positiveInteger"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
</xs:element>
 </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Anexo II:


```
<?xml version="1.0" encoding="UTF-8"?>
<busqueda xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="xml/estructuraResultados.xsd">
 <parametros>
  <objetivo>twitter-flickr-youtube</objetivo>
  <key>uc3m</key>
  <rpp>15</rpp>
  <pag>1</pag>
 </parametros>
 <resultados>
  <resultado>
   <redsocial>Twitter</redsocial>
   <autor>jberlana</autor>
   <titulo>Tweet del usuario 'jberlana', 2010-05-17 a las 12:11:12.</titulo>
   <descripcion>Poco ambiente de estudio en la biblioteca de la #uc3m.</descripcion>
   <informacion>Poco ambiente de estudio en la biblioteca de la #uc3m.</informacion>
   <fecha>2010-05-17</fecha>
   <hora>12:11:12</hora>
   <contenido>TEXTO</contenido>
   <thumb>http://a1.twimg.com/profile_images/389745954/Captura_de_pantalla_2009-08-
30_a_las_20.18.34_normal.png</thumb>
   <etiquetas>
    <etiqueta>uc3m</etiqueta>
   </etiquetas>
  </resultado>
  <resultado>
   <redsocial>Twitter</redsocial>
   <autor>biblioteca_uc3m</autor>
   <titulo>Tweet del usuario 'biblioteca_uc3m', 2010-05-17 a las 10:43:42.</titulo>
   <descripcion>En el #Diadeinternet , Â¿por quÃ© no echas un vistazo a todo lo que ofrece nuestro
portal Web?: http://biblioteca.uc3m.es</descripcion>
   <informacion>En el #Diadeinternet , Â¿por quÃ© no echas un vistazo a todo lo que ofrece nuestro
portal Web?: http://biblioteca.uc3m.es</informacion>
   <fecha>2010-05-17</fecha>
   <hora>10:43:42</hora>
   <contenido>TEXTO</contenido>
   <thumb>http://a1.twimg.com/profile_images/686950350/Bandafacebook_normal.jpg</thumb>
   <etiquetas>
    <etiqueta>Diadeinternet</etiqueta>
   </etiquetas>
  </resultado>
  <resultado>
   <redsocial>Twitter</redsocial>
   <autor>barrogante</autor>
   <titulo>Tweet del usuario 'barrogante', 2010-05-17 a las 09:47:01.</titulo>
   <descripcion>Conferencia sobre la reforma constitucional: Leonel FernÃ¡ndez Presidente de
RepÃºblica Dominicana. Hoy 18:15 h Aula Magna Campus Getafe #uc3m</descripcion>
   <informacion>Conferencia sobre la reforma constitucional: Leonel FernÃ¡ndez Presidente de
RepÃºblica Dominicana. Hoy 18:15 h Aula Magna Campus Getafe #uc3m</informacion>
   <fecha>2010-05-17</fecha>
   <hora>09:47:01</hora>
   <contenido>TEXTO</contenido>
   <thumb>http://a3.twimg.com/profile_images/719181365/Foto_Bel_n1_normal.jpg</thumb>
   <etiquetas>
    <etiqueta>uc3m</etiqueta>
```

```
    </etiquetas>
   </resultado>
   <resultado>
    <redsocial>Twitter</redsocial>
    <autor>poppiefresh</autor>
    <titulo>Tweet del usuario 'poppiefresh', 2010-05-17 a las 01:32:17.</titulo>
    <descripcion>&amp;quot;I donÂ´t need sex, uc3m fucks me every single day&amp;quot;
AMÃ‰N</descripcion>
    <informacion>&amp;quot;I donÂ´t need sex, uc3m fucks me every single day&amp;quot;
AMÃ‰N</informacion>
    <fecha>2010-05-17</fecha>
    <hora>01:32:17</hora>
    <contenido>TEXTO</contenido>
    <thumb>http://a3.twimg.com/profile_images/684047241/DSC_0163_2_normal.jpg</thumb>
   </resultado>
   <resultado>
    <redsocial>Twitter</redsocial>
    <autor>madrimasd</autor>
    <titulo>Tweet del usuario 'madrimasd', 2010-05-17 a las 00:50:28.</titulo>
    <descripcion>Una moto de competiciÃ³n hecha por estudiantes: Investigadores de la Universidad
Carlos III de Madrid (UC3M) han pr... http://bit.ly/bHfTVN</descripcion>
    <informacion>Una moto de competiciÃ³n hecha por estudiantes: Investigadores de la Universidad
Carlos III de Madrid (UC3M) han pr... http://bit.ly/bHfTVN</informacion>
    <fecha>2010-05-17</fecha>
    <hora>00:50:28</hora>
    <contenido>TEXTO</contenido>
    <thumb>http://a1.twimg.com/profile_images/23991722/mi_d_90x90_normal.gif</thumb>
   </resultado>
   <total>
    <red>Twitter</red>
    <numresult>5</numresult>
    <pagina>1</pagina>
   </total>
  </resultados>
  <resultados>
   <resultado>
    <redsocial>Flickr</redsocial>
    <autor>emadridnet</autor>
    <titulo>Seminario eMadrid sobre â□œAprendizaje 3Dâ□□ (2010-03-12)</titulo>
    <descripcion>Seminario eMadrid sobre â□œAprendizaje 3Dâ□□
Universidad Carlos III de Madrid (Campus de LeganÃ©s)
2010-03-12
&lt;a href="http://emadridnet.org" rel="nofollow"&gt;emadridnet.org&lt;/a&gt;
Carlos Delgado Kloos, PresentaciÃ³n grupo GAST (UC3M)
IMG_2100</descripcion>
    <informacion>http://farm4.static.flickr.com/3408/4598635656_23332f8502.jpg</informacion>
    <fecha>2010-05-11</fecha>
    <hora>15:24:14</hora>
    <contenido>IMAGEN</contenido>
    <thumb>http://farm4.static.flickr.com/3408/4598635656_23332f8502_s.jpg</thumb>
    <etiquetas>
     <etiqueta>seminario</etiqueta>
     <etiqueta>emadrid</etiqueta>
     <etiqueta>emadridnet</etiqueta>
     <etiqueta>aprendizaje3d</etiqueta>
     <etiqueta>uc3m</etiqueta>
     <etiqueta>gast</etiqueta>
```

```
    </etiquetas>
    <extra>
     <item name="Fotógrafo">eMadrid Net</item>
     <item name="Fecha Captura">2010-03-12 15:08:59</item>
    </extra>
   </resultado>
   <resultado>
    <redsocial>Flickr</redsocial>
    <autor>emadridnet</autor>
    <titulo>Seminario eMadrid sobre â□œAprendizaje 3Dâ□□ (2010-03-12)</titulo>
    <descripcion>Seminario eMadrid sobre â□œAprendizaje 3Dâ□□
Universidad Carlos III de Madrid (Campus de LeganÃ©s)
2010-03-12
&lt;a href="http://emadridnet.org" rel="nofollow"&gt;emadridnet.org&lt;/a&gt;
MÂº Blanca IbÃ¡Ã±ez, grupo GAST (UC3M)
IMG_2107</descripcion>
    <informacion>http://farm4.static.flickr.com/3339/4598018887_9165af1947.jpg</informacion>
    <fecha>2010-05-11</fecha>
    <hora>15:24:21</hora>
    <contenido>IMAGEN</contenido>
    <thumb>http://farm4.static.flickr.com/3339/4598018887_9165af1947_s.jpg</thumb>
    <etiquetas>
     <etiqueta>seminario</etiqueta>
     <etiqueta>emadrid</etiqueta>
     <etiqueta>emadridnet</etiqueta>
     <etiqueta>aprendizaje3d</etiqueta>
     <etiqueta>uc3m</etiqueta>
     <etiqueta>gast</etiqueta>
    </etiquetas>
    <extra>
     <item name="Fotógrafo">eMadrid Net</item>
     <item name="Fecha Captura">2010-03-12 15:39:34</item>
    </extra>
   </resultado>
   <resultado>
    <redsocial>Flickr</redsocial>
    <autor>emadridnet</autor>
    <titulo>Seminario eMadrid sobre â□œAprendizaje 3Dâ□□ (2010-03-12)</titulo>
    <descripcion>Seminario eMadrid sobre â□œAprendizaje 3Dâ□□
Universidad Carlos III de Madrid (Campus de LeganÃ©s)
2010-03-12
&lt;a href="http://emadridnet.org" rel="nofollow"&gt;emadridnet.org&lt;/a&gt;
Carlos Delgado Kloos, PresentaciÃ³n grupo GAST (UC3M)
IMG_2101</descripcion>
    <informacion>http://farm4.static.flickr.com/3398/4598635570_e149e8d355.jpg</informacion>
    <fecha>2010-05-11</fecha>
    <hora>15:24:12</hora>
    <contenido>IMAGEN</contenido>
    <thumb>http://farm4.static.flickr.com/3398/4598635570_e149e8d355_s.jpg</thumb>
    <etiquetas>
     <etiqueta>seminario</etiqueta>
     <etiqueta>emadrid</etiqueta>
     <etiqueta>emadridnet</etiqueta>
     <etiqueta>aprendizaje3d</etiqueta>
     <etiqueta>uc3m</etiqueta>
     <etiqueta>gast</etiqueta>
    </etiquetas>
```

```
  <extra>
   <item name="Fotógrafo">eMadrid Net</item>
   <item name="Fecha Captura">2010-03-12 15:09:08</item>
  </extra>
 </resultado>
 <resultado>
  <redsocial>Flickr</redsocial>
  <autor>javialamo</autor>
  <titulo>UC3M Train</titulo>
  <descripcion/>
  <informacion>http://farm5.static.flickr.com/4014/4589091213_d992b8f448.jpg</informacion>
  <fecha>2010-05-08</fecha>
  <hora>19:48:04</hora>
  <contenido>IMAGEN</contenido>
  <thumb>http://farm5.static.flickr.com/4014/4589091213_d992b8f448_s.jpg</thumb>
  <etiquetas>
   <etiqueta>train</etiqueta>
   <etiqueta>uc3m</etiqueta>
   <etiqueta>leganÃ©s</etiqueta>
   <etiqueta>dagd60project</etiqueta>
  </etiquetas>
  <extra>
   <item name="Fotógrafo">Javier del Ã□lamo</item>
   <item name="Fecha Captura">2010-05-08 19:48:04</item>
  </extra>
 </resultado>
 <resultado>
  <redsocial>Flickr</redsocial>
  <autor>javialamo</autor>
  <titulo>Bajo el tren</titulo>
  <descripcion/>
  <informacion>http://farm5.static.flickr.com/4039/4589091221_928889ca34.jpg</informacion>
  <fecha>2010-05-08</fecha>
  <hora>19:48:04</hora>
  <contenido>IMAGEN</contenido>
  <thumb>http://farm5.static.flickr.com/4039/4589091221_928889ca34_s.jpg</thumb>
  <etiquetas>
   <etiqueta>train</etiqueta>
   <etiqueta>rails</etiqueta>
   <etiqueta>uc3m</etiqueta>
   <etiqueta>leganÃ©s</etiqueta>
   <etiqueta>dagd60project</etiqueta>
  </etiquetas>
  <extra>
   <item name="Fotógrafo">Javier del Ã□lamo</item>
   <item name="Fecha Captura">2010-05-08 19:48:04</item>
  </extra>
 </resultado>
 <total>
  <red>Flickr</red>
  <numresult>5</numresult>
  <pagina>1</pagina>
  <totresult>1660</totresult>
  <totpaginas>332</totpaginas>
 </total>
</resultados>
<resultados>
```

```
  <resultado>
    <redsocial>Youtube</redsocial>
    <autor>losliriums</autor>
    <titulo>Frame Videoclip Uc3m</titulo>
    <descripcion>LETRA DE FRAME Baby mÃ-rale, dime que es lo que ves, mejor yo te lo dirÃ© estÃ¡s viendo a un director que es novel. Pero que va a triunfar en BerlÃ-n o en Cannes. Capaz de rodar Rambo tres con el ritmo de Isabel Croixet. Frame Cuarenta y ocho Kilo herzios para el sonido grabar. Frame No uses el cuatro tercios que no es profesional. Frame financiaciÃ³n de un banco sabemos que no tendrÃ¡s. Frame haz el balance de blancos puedes tu cara usar Rendeando Rendeando Rendeando Rendeando HarÃ¡ un videoclip estilo David Lynch que nadie comprenderÃ¡, por eso le alabarÃ¡n. Y es verdad que tiene que aprender dos cosillas o tres, o tres millones que mÃ¡s da. Â¡Venga enseÃ±ad! Usa el manual. Frame Lleva gafas de pasta es un intelectual Frame Pero con eso no basta te falta mucha humildad Rendeando, Rendeando, Rendeando, Rendeando Frame Has puesto mal el formato, lo tienes en DVCAM Fame Crees que has pasado un mal rato, aÃºn te queda montar Frame Dejad de tocar botones yo soy aquÃ- el director Frame No se ve nada cojones tienes que abrir el visor Rendeando Rendeando Rendeando Rendeando</descripcion>
    <informacion>http://www.youtube.com/v/KpqStNTlCMg</informacion>
    <fecha>2010-05-14</fecha>
    <hora>17:27:00</hora>
    <contenido>VIDEO</contenido>
    <extension>3gp</extension>
    <thumb>http://i.ytimg.com/vi/KpqStNTlCMg/default.jpg</thumb>
    <etiquetas>
      <etiqueta>Frame</etiqueta>
      <etiqueta>Videoclip</etiqueta>
      <etiqueta>uc3m</etiqueta>
    </etiquetas>
    <extra>
      <item name="DuraciÃ³n">174</item>
      <item name="Reproducciones">40</item>
    </extra>
  </resultado>
  <resultado>
    <redsocial>Youtube</redsocial>
    <autor>Yulia0rigliasso</autor>
    <titulo>Observados (Trailer)</titulo>
    <descripcion>Trailer del cortometraje 'Observados', realizado para la asignatura de taller de televisiÃ³n en la UC3M. La canciÃ³n del trailer es 'Camera', del grupo Plastiscines. ACTORES: Noelia MartÃ-nez Lorenzo Raul Ãlvarez BelÃ©n de Santiago ChÃ©rcoles Marisol Pascual Carabella Clara Bahamonde Matas Silvia Fidalgo Dominguez Y la sÃºper colaboraciÃ³n de Bely!</descripcion>
    <informacion>http://www.youtube.com/v/D2lQV2unMqI</informacion>
    <fecha>2010-05-14</fecha>
    <hora>13:31:19</hora>
    <contenido>VIDEO</contenido>
    <extension>3gp</extension>
    <thumb>http://i.ytimg.com/vi/D2lQV2unMqI/default.jpg</thumb>
    <etiquetas>
      <etiqueta>corto</etiqueta>
      <etiqueta>observados</etiqueta>
      <etiqueta>espejo</etiqueta>
      <etiqueta>trailer</etiqueta>
      <etiqueta>uc3m</etiqueta>
      <etiqueta>grupo</etiqueta>
      <etiqueta>72</etiqueta>
      <etiqueta>audiovisual</etiqueta>
      <etiqueta>scream</etiqueta>
      <etiqueta>plastiscines</etiqueta>
```

```
      <etiqueta>camera</etiqueta>
      <etiqueta>2010</etiqueta>
      <etiqueta>taller</etiqueta>
      <etiqueta>television</etiqueta>
      <etiqueta>cortometraje</etiqueta>
      <etiqueta>colmenarejo</etiqueta>
      <etiqueta>belen</etiqueta>
      <etiqueta>santiago</etiqueta>
      <etiqueta>cheroles</etiqueta>
      <etiqueta>raul</etiqueta>
      <etiqueta>alvarez</etiqueta>
      <etiqueta>noelia</etiqueta>
      <etiqueta>martinez</etiqueta>
      <etiqueta>lorenzo</etiqueta>
    </etiquetas>
    <extra>
      <item name="DuraciÃ³n">62</item>
      <item name="NÃºmero de Votos">1</item>
      <item name="ValoraciÃ³n">5</item>
      <item name="Reproducciones">58</item>
    </extra>
  </resultado>
  <resultado>
    <redsocial>Youtube</redsocial>
    <autor>kelitaWC</autor>
    <titulo>A dream comes true. videoclip uc3m</titulo>
    <descripcion>Videoclip realizado por LucÃ-a, Yasmina, Eugenia, Ana, Rafa y Kela, alumnos de la
Universidad Carlos III de Madrid. … canciÃ³n: SOMEWHERE OVER THE RAINBOW de Norah
Jones.</descripcion>
    <informacion>http://www.youtube.com/v/s3QKiXVSoYI</informacion>
    <fecha>2010-05-14</fecha>
    <hora>12:51:34</hora>
    <contenido>VIDEO</contenido>
    <extension>3gp</extension>
    <thumb>http://i.ytimg.com/vi/s3QKiXVSoYI/default.jpg</thumb>
    <etiquetas>
      <etiqueta>uc3m</etiqueta>
      <etiqueta>videoclip</etiqueta>
      <etiqueta>mendiguillo</etiqueta>
      <etiqueta>so payaso</etiqueta>
      <etiqueta>marionetas</etiqueta>
      <etiqueta>somewhere over the rainbow</etiqueta>
    </etiquetas>
    <extra>
      <item name="DuraciÃ³n">100</item>
      <item name="Reproducciones">4</item>
    </extra>
  </resultado>
  <resultado>
    <redsocial>Youtube</redsocial>
    <autor>billycompany</autor>
    <titulo>The Raveonettes - LUST (uc3m)</titulo>
    <descripcion>TÃ©cnicas y tecnologÃ-as del Audiovisual - 2010 The Raveonettes - Lust (todos los
derechos reservados al grupo) por Ana PÃ©rez LÃ³pez Antonio SuÃ¡rez-Bustamante Anaya Juan Javier
Torres Faba Sara Valbuena LobatÃ³n y David Villafranca PemÃ¡n Universidad Carlos III de Madrid
(uc3m)</descripcion>
    <informacion>http://www.youtube.com/v/DHAPNSUe8jw</informacion>
```

```
    <fecha>2010-05-14</fecha>
    <hora>08:40:10</hora>
    <contenido>VIDEO</contenido>
    <extension>3gp</extension>
    <thumb>http://i.ytimg.com/vi/DHAPNSUe8jw/default.jpg</thumb>
    <etiquetas>
     <etiqueta>videoclip</etiqueta>
     <etiqueta>uc3m</etiqueta>
     <etiqueta>video</etiqueta>
     <etiqueta>clip</etiqueta>
     <etiqueta>raveonettes</etiqueta>
     <etiqueta>lust</etiqueta>
    </etiquetas>
    <extra>
     <item name="DuraciÃ³n">149</item>
     <item name="Reproducciones">108</item>
    </extra>
   </resultado>
   <resultado>
    <redsocial>Youtube</redsocial>
    <autor>UC3M</autor>
    <titulo>Una moto de competiciÃ³n hecha por estudiantes</titulo>
    <descripcion>Investigadores de la Universidad Carlos III de Madrid (UC3M) han presentado el
Proyecto Moto Student, en el que un grupo de estudiantes e ingenieros de la instituciÃ³n trabajan para
diseÃ±ar y desarrollar un prototipo de motocicleta de competiciÃ³n que competirÃ¡ contra equipos
universitarios de todo el mundo.</descripcion>
    <informacion>http://www.youtube.com/v/tHWN2AexnZQ</informacion>
    <fecha>2010-05-13</fecha>
    <hora>13:52:36</hora>
    <contenido>VIDEO</contenido>
    <extension>3gp</extension>
    <thumb>http://i.ytimg.com/vi/tHWN2AexnZQ/default.jpg</thumb>
    <etiquetas>
     <etiqueta>UC3M universidad</etiqueta>
     <etiqueta>Moto Student</etiqueta>
     <etiqueta>Moto MAQLAB</etiqueta>
    </etiquetas>
    <extra>
     <item name="DuraciÃ³n">125</item>
     <item name="Reproducciones">1</item>
    </extra>
   </resultado>
   <total>
    <red>Youtube</red>
    <numresult>5</numresult>
    <pagina>1</pagina>
    <totresult>1340</totresult>
    <totpaginas>268</totpaginas>
   </total>
  </resultados>
</busqueda>
```

13.- Bibliografía y referencias

-Libros

[1] Ian Sommerville, 'Ingeniería del Software' (Prentice Hall, 2009, 7ª. Edición).

[2] Holzner, Steven ,'PHP 5 El lenguaje para los profesionales de la Web' (Anaya)

[3] John Coggeshal 'PHP 5 Developer's Handbook' (Sams)

[4] David Flanagan, 'Javascript (la guía definitiva)' (Anaya)

[5] Andy Budd, 'CSS Mastery: Advanced Web Standards Solutions' (Friendsof)

-Referencias en Internet

[6] "Measuring Tweets". Twitter Blog del 22 de Febrero, 2010. Disponible en http://blog.twitter.com/2010/02/measuring-tweets.html (accedido en Junio, 2010)

[7] "By the numbers: Twitter Vs. Facebook Vs. Google Buzz". Search Engine Land con fecha 23 de Febrero, 2010. Disponible en http://searchengineland.com/by-the-numbers-twitter-vs-facebook-vs-google-buzz-36709 (accedido en Junio, 2010)

[8] eXtensible Markup Language – especificación. Disponible en: http://www.w3.org/XML/ (accedido en Enero, 2010)

[9] Real Decreto 4/2010, de 8 de enero, Regulación de El Esquema de Interoperabilidad en el ámbito de la Administración Pública. Boletín Oficial del Estado BOE-A-2010-1331, Boletín Núm. 25, del 29 de enero de 2010, Sec. I Pág. 8139.

[10] Youtube Site Info – Disponible en: http://www.alexa.com/siteinfo/youtube.com (accedido en Junio 2010).

[11] Model-View-Controller – Disponible en: http://msdn.microsoft.com/en-us/library/ff649643.aspx (accedido en Abril 2010).

[12] Ilustración MVC – Disponible en http://www.symfony-project.org (último acceso en Junio 2010).

[13] Estadísticas de Tumblr – Disponible en http://www.tumblr.com/about (último acceso en Junio 2010).

[14] eXtensible Markup Language – descripción. Disponible en: http://www.w3schools.com/xml (accedido en Febrero 2010).

[15] XSL Transformations – especificación. Disponible en http://www.w3.org/TR/xslt (ultimo acceso en Junio 2010).

[16] XSL Transformations – descripción. Disponible en http://www.w3schools.com/xsl (accedido en Junio 2010).

[17] Document Type Definition – descripción. Disponible en http://www.w3schools.com/dtd (accedido en Febrero 2010).

[18] XML Schema – especificación. Disponible en http://www.w3.org/XML/Schema.html (último acceso en Junio 2010).

[19] XML Schema – descripción. Disponible en http://www.w3schools.com/schema (accedido en Enero 2010).

[20] Xpath – especificación. Disponible en http://www.w3.org/TR/xpath (accedido en Enero 2010).

[21] Xpath – descripción. Disponible en http://www.w3schools.com/xpath (accedido en Enero 2010).

[22] Xquery – especificación. Disponible en http://www.w3.org/TR/xquery (accedido en Enero 2010).

[23] Xquery – descripción. Disponible en http://www.w3schools.com/rss (accedido en Enero 2010).

[24] RSS – descripción. Disponible en http://www.w3schools.com/xquery (accedido en Febrero 2010).

[25] Atom – especificación. Disponible en http://www.w3.org/2005/Atom.html (accedido en Febrero 2010).

[26] Json – descripción. Disponible en http://www.json.org/ (accedido en Febrero 2010).

[27] Web - Services especificación. Disponible en http://www.w3.org/2002/ws/ (accedido en Enero 2010).

[28] SOAP - especificación. Disponible en http://www.w3.org/TR/soap (accedido en Febrero 2010).

[29] REST - descripción. Disponible en http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (accedido en Enero 2010).

[30] PHP – Disponible en: http://www.php.net (accedido en Enero 2010)

[31] SAX - Disponible en: http://php.net/manual/es/book.xml.php (accedido en Febrero 2010).

[32] DOM - Disponible en: http://www.php.net/manual/es/book.dom.php (Accedido en Febrero 2010).

[33] SimpleXML - Disponible en: http://www.php.net/manual/es/book.simplexml.php (accedido en Febrero 2010).

[34] Simple API for XML – Disponible en: http://www.saxproject.org/ (accedido en Marzo 2010)